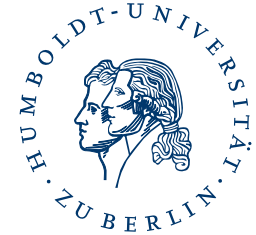
Kryptologie

Johannes Köbler



Institut für Informatik
Humboldt-Universität zu Berlin

WS 2020/21

Organisatorisches

Aktuelle Infos auf der VL-Webseite unter

- *https://hu.berlin/vlkrypto*

bzw.

- *https://www.informatik.hu-berlin.de/de/forschung/gebiete/algorithmenII/Lehre/ws20/krypto*

Ablauf

Skript, Folien und Aufgabenblätter

- Skript, Folien und Aufzeichnung werden jeweils nach der Vorlesung ins Netz (Webseite bzw. Moodle) gestellt
- Übungsblätter werden in der Regel dienstags veröffentlicht
- Die Besprechung der mündlichen Aufgaben erfolgt am Freitag der Folgewoche. Lösungen dazu können bis zum Tag davor in Moodle hochgeladen werden, Details siehe dort
- Die schriftlichen Aufgaben sind bis Dienstag zwei Wochen nach Ausgabe um 23:59 Uhr abzugeben
- Fragen zu Übung und Vorlesung können im Moodle-Forum auch asynchron gestellt und diskutiert werden

Übungen

Anmeldung

- über Agnes
- und bei Moodle (wegen Punktevergabe und Bildung von Abgabegruppen)
- Mails von Agnes und von Moodle werden standardmäßig an den HU-Account gesendet (bitte regelmäßig checken)

Ausgabe der Aufgabenblätter

- über Moodle und auf der VL-Webseite

Abgabe von Lösungen

- digital über Moodle

Bearbeitung der Übungsaufgaben

- in Gruppen von bis zu drei Teilnehmern
- Lösungen für die schriftlichen Aufgaben sollten als PDF abgegeben werden
- die Abgabe von Lösungsvorschlägen für die mündlichen Aufgaben ist freiwillig und geht nicht in die Punktewertung ein
- Lösungsvorschläge für die mündlichen Aufgaben können auch per Texteingabe gemacht werden
- besonders gut gelungene Lösungen werden mit Zustimmung der/des Abgebenden im Forum veröffentlicht

Übungsschein und Prüfung

Scheinkriterien

- Lösen von mindestens 50% der schriftlichen Aufgaben

Prüfungsform

- voraussichtlich mündlich
- Der Übungsschein ist ***nicht*** Prüfungsvoraussetzung

Gibt es zum organisatorischen Ablauf noch Fragen?

Lernziele

- Kryptografische Verfahren schaffen Vertrauen in ungeschützten Umgebungen
- Sie ermöglichen sichere Kommunikation über unsichere Kanäle und können verhindern, dass sich ein Kommunikationspartner unfair verhält
- In unsicheren Umgebungen wie dem Internet können sie die aus direkter Interaktion gewohnte Sicherheit herstellen
- Und auch die Interaktion in sicheren Umgebungen wird um Möglichkeiten erweitert, die ohne Kryptografie nicht denkbar wären
- Im Bachelormodul ***Einführung in die Kryptologie*** haben wir uns mit den mathematischen Grundlagen von kryptografischen Verfahren beschäftigen, wobei (symmetrische und asymmetrische) Verschlüsselungsverfahren im Vordergrund standen
- Im aktuellen Mastermodul ***Kryptologie*** werden wir dagegen kryptografische Verfahren und Protokolle für andere Schutzziele betrachten wie z.B. Hashverfahren und digitale Signaturen sowie Pseudozufallsgeneratoren

Kryptosysteme

- Kryptosysteme (Verschlüsselungsverfahren) dienen der Geheimhaltung von Nachrichten bzw. Daten
- Hierzu gibt es auch andere Methoden wie z.B.
  - Physikalische Maßnahmen: Tresor etc.
  - Organisatorische Maßnahmen: einsamer Waldspaziergang etc.
  - Steganografische Maßnahmen: unsichtbare Tinte etc.

Überblick weiterer Schutzziele

Andererseits können durch kryptografische Verfahren weitere Schutzziele realisiert werden wie z.B.

- Vertraulichkeit
  - Geheimhaltung
  - Anonymität (z.B. Mobiltelefon)
  - Unbeobachtbarkeit (von Transaktionen)
- Integrität
  - von Nachrichten und Daten
- Zurechenbarkeit
  - Authentikation
  - Unabstreitbarkeit
  - Identifizierung
- Verfügbarkeit
  - von Daten
  - von Rechenressourcen
  - von Informationsdienstleistungen

Kryptologie

In das Umfeld der Kryptologie fallen die folgenden Begriffe

- Kryptografie:
  Lehre von der Geheimhaltung von Informationen durch Verschlüsselung
  Im weiteren Sinne: Wissenschaft von der Übermittlung, Speicherung und Verarbeitung von Daten in einer von potentiellen Gegnern bedrohten Umgebung
- Kryptoanalysis:
  Erforschung der Methoden eines unbefugten Angriffs gegen ein Kryptoverfahren
  Zweck: Vereitelung der mit seinem Einsatz verfolgten Ziele
- Kryptoanalyse:
  Analyse eines Kryptoverfahrens zum Zweck der Bewertung seiner kryptografischen Stärken und Schwächen
- Kryptologie:
  Wissenschaft vom Entwurf, der Anwendung und der Analyse von kryptografischen Verfahren (umfasst Kryptografie und Kryptoanalyse)

Kryptografische Hashverfahren

- sind ein wirksames Werkzeug zur Sicherstellung der Integrität von Nachrichten oder generell von digitalisierten Daten
- Sie nehmen somit beim Schutz der Datenintegrität eine ähnlich herausragende Stellung ein wie sie Kryptosystemen bei der Wahrung der Vertraulichkeit zukommt
- Daneben finden kryptografische Hashfunktionen aber auch vielfach als Bausteine von komplexeren Systemen Verwendung
- Wie wir noch sehen werden, sind kryptografische Hashfunktionen etwa bei der Erstellung von digitalen Signaturen sehr nützlich
- Auf weitere Anwendungsmöglichkeiten werden wir später eingehen

Kryptografische Hashverfahren

- Vielen Anwendungen von kryptografischen Hashfunktionen h liegt die Idee zugrunde, dass sie zu einem vorgegebenen Text x eine zwar kompakte aber dennoch repräsentative Darstellung $h(x)$ liefern, die unter praktischen Gesichtspunkten als eine eindeutige Identifikationsnummer von x fungieren kann
- Die Berechnungsvorschrift für h muss somit „charakteristische Merkmale“ von x in den Hashwert $h(x)$ einfließen lassen
- Da der Fingerabdruck eines Menschen ganz ähnliche Eigenschaften besitzt (was ihn für Kriminalisten bekanntlich so wertvoll macht), wird der Hashwert $h(x)$ auch oft als ein ***digitaler Fingerabdruck*** von x bezeichnet
- Gebräuchlich sind auch die Bezeichnungen kryptografische Prüfsumme oder message digest (englische Bezeichnung für „Nachrichtenextrakt“)

Kryptografische Hashverfahren

Typische Schutzziele, die sich mittels Hashfunktionen realisieren lassen:

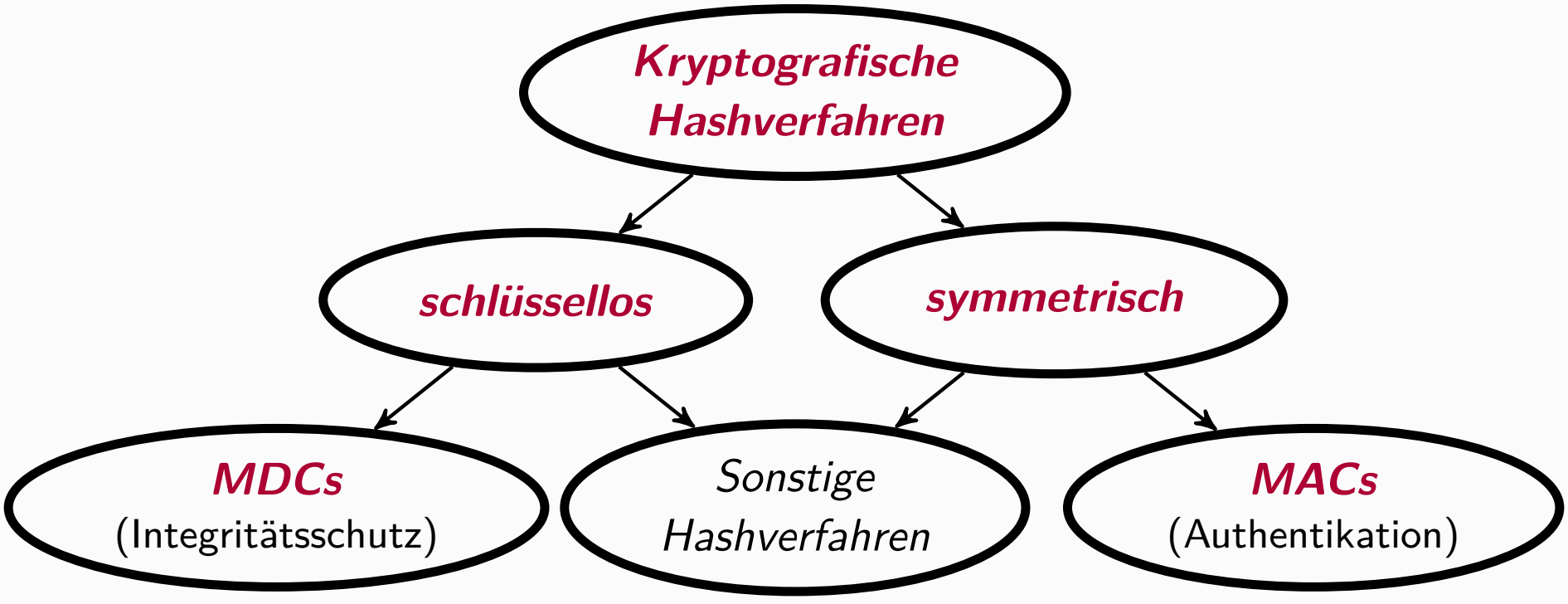
Nachrichtenauthentikation (message authentication)

- Wie lässt sich sicherstellen, dass eine Nachricht (oder eine Datei) während einer (räumlichen oder auch zeitlichen) Übertragung nicht verändert wurde?
- Wie lässt sich der Urheber (oder Absender) einer Nachricht zweifelsfrei feststellen?

Teilnehmerauthentikation (entity authentication, identification)

- Wie kann sich eine Person (oder ein Gerät) anderen gegenüber zweifelsfrei ausweisen?

Klassifikation von Hashverfahren

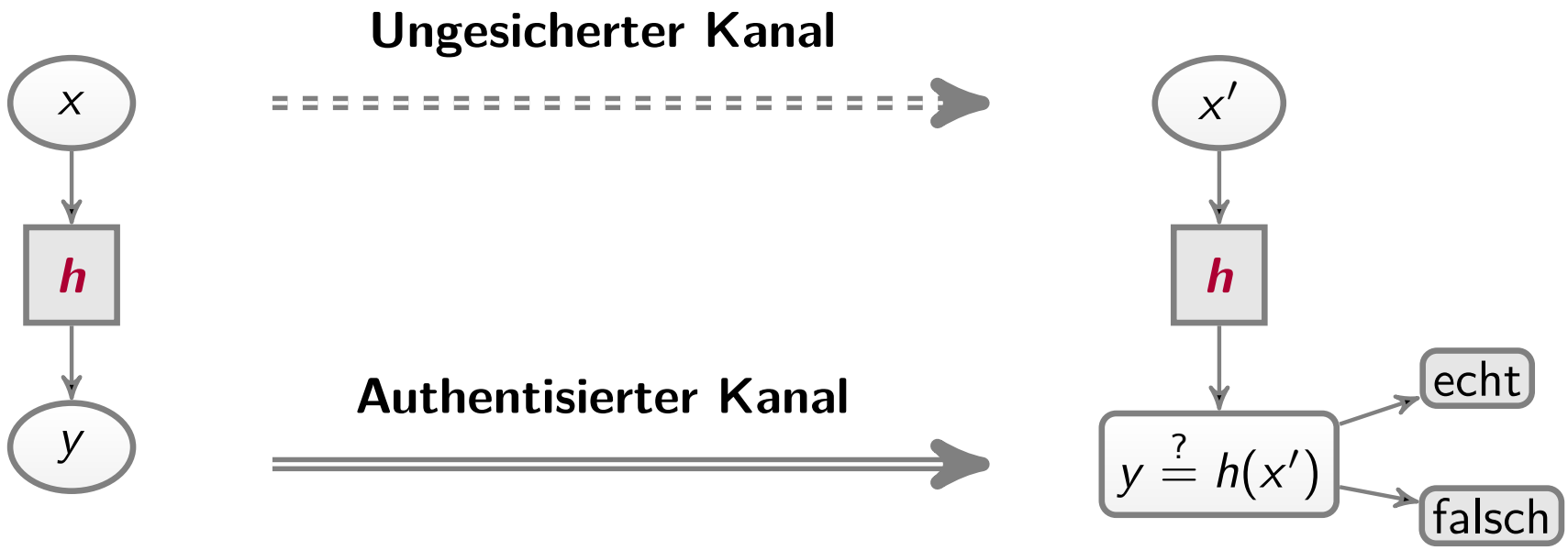


Kryptografische Hashverfahren lassen sich grob danach klassifizieren, ob der Hashwert lediglich in Abhängigkeit vom Eingabetext berechnet wird oder zusätzlich von einem symmetrischen Schlüssel abhängt

Schlüssellose Hashfunktionen

- Kryptografische Hashfunktionen, bei deren Berechnung keine Schlüssel benutzt werden, dienen vornehmlich der Erkennung von unbefugt vorgenommenen Manipulationen an Dateien oder Nachrichten
- Daher werden sie auch als ***MDC*** (***M****anipulation* ***D****etection* ***C****ode*) bezeichnet
- Zuweilen wird das Kürzel ***MDC*** auch als eine Abkürzung für ***M****odification* ***D****etection* ***C****ode* verwendet
- Seltener ist dagegen die Bezeichnung ***MIC*** (***m****essage* ***i****ntegrity* ***c****odes*)

Manipulation Detection Codes



Um die Integrität eines Datensatzes x sicherzustellen, der über einen ungesicherten Kanal gesendet (bzw. auf einem vor Manipulationen nicht sicheren Webserver abgelegt) wird, kann man wie folgt verfahren

- Der MDC-Hashwert $y = h(x)$ von x wird auf einem authentisierten Kanal übertragen
- Nach der Übertragung wird geprüft, ob der Datensatz noch den Hashwert y liefert

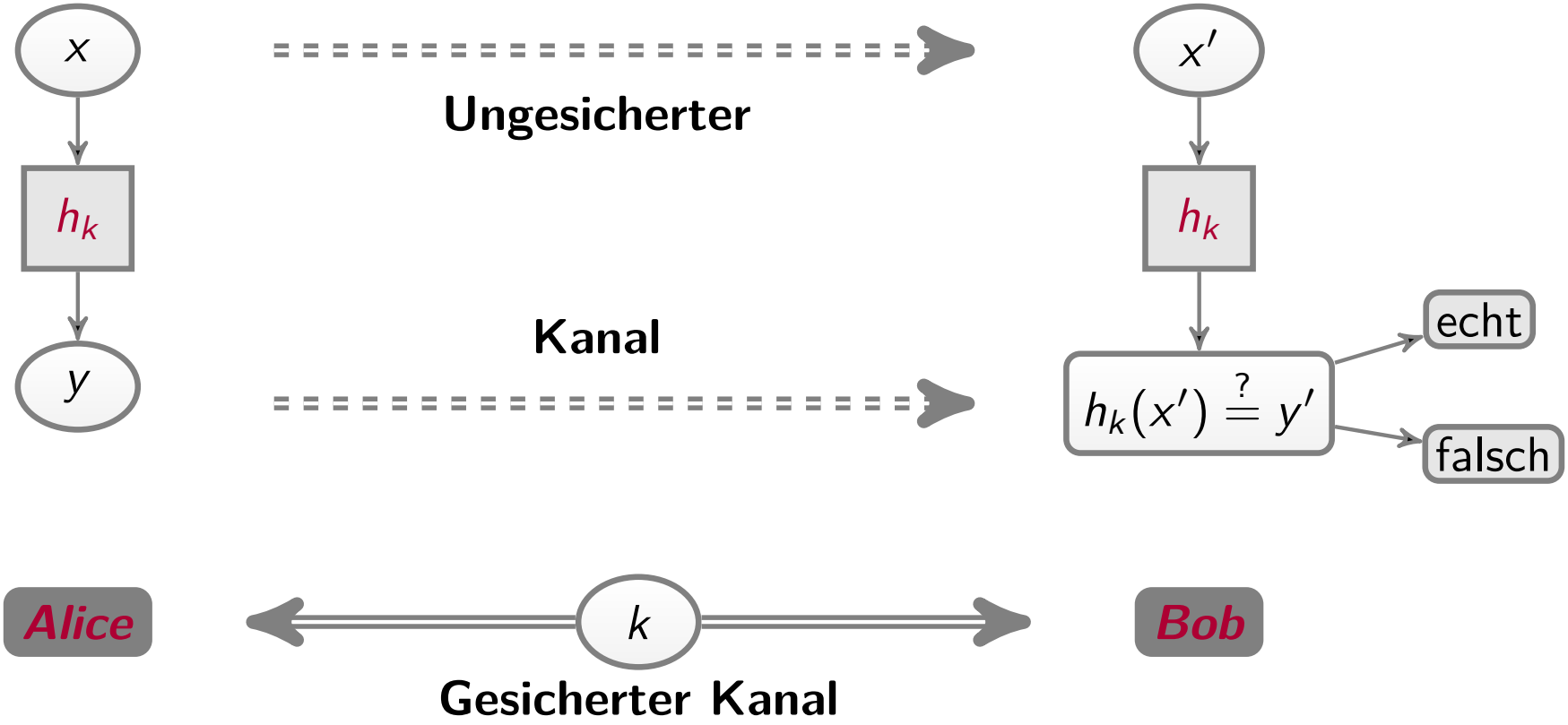
Hashverfahren mit symmetrischen Schlüsseln

- Kryptografische Hashverfahren mit symmetrischen Schlüsseln finden hauptsächlich bei der Authentifizierung von Nachrichten Verwendung
- Diese werden daher auch als ***MAC*** (***m****essage* ***a****uthentication* ***c****ode*) oder als ***Authentikationscode*** bezeichnet
- Daneben gibt es auch Hashverfahren mit asymmetrischen Schlüsseln
- Diese werden jedoch der Rubrik der Signaturverfahren zugeordnet, da mit ihnen ausschließlich digitale Signaturen gebildet werden

Hashverfahren mit symmetrischen Schlüsseln

- Die Abbildung auf der nächsten Folie zeigt, wie sich Nachrichten mit einem MAC authentisieren lassen
- Man beachte, dass nun auch der Hashwert über den unsicheren Kanal gesendet wird
- Möchte Alice eine Nachricht x an Bob übermitteln, so berechnet sie den zugehörigen MAC-Wert $y = h_k(x)$ und fügt diesen der Nachricht x hinzu
- Bob überprüft die Echtheit der empfangenen Nachricht (x', y'), indem er seinerseits den zu x' gehörigen Hashwert $h_k(x')$ berechnet und das Ergebnis mit y' vergleicht
- Der geheime Authentikationsschlüssel k muss hierbei genau wie bei einem symmetrischen Kryptosystem über einen gesicherten Kanal vereinbart werden

Verwendung eines MAC zur Nachrichtenauthentikation



- Hierbei ist k der symmetrische Authentikationsschlüssel und $y = h_k(x)$ der MAC-Wert für x unter k
- Indem Alice ihre Nachricht x um den Hashwert $y = h_k(x)$ ergänzt, hat Bob nicht nur die Möglichkeit, anhand von y die empfangene Nachricht x' auf Manipulationen, sondern auch ihre Herkunft zu überprüfen

Schlüssellose Hashfunktionen (MDCs)

- Wir betrachten nun verschiedene Sicherheitsanforderungen an MDCs h
- Dabei nehmen wir an, dass $h\colon X \to Y$ öffentlich bekannt ist
- Ein Paar $(x, y) \in X \times Y$ heißt ***gültig*** für h, falls $h(x) = y$ ist
- Ein Paar (x, x') mit $x \neq x'$ und $h(x) = h(x')$ heißt ***Kollisionspaar*** für h
- Die Anzahl $\|Y\|$ der Hashwerte bezeichnen wir mit m
- Ist auch der Textraum X endlich, $\|X\| = n$, so heißt h eine (n, m)-***Hashfunktion***
- In diesem Fall verlangen wir meist, dass $n \geq 2m$ ist, und wir nennen h dann eine ***Kompressionsfunktion*** (compression function)

Schlüssellose Hashfunktionen (MDCs)

- Da h öffentlich bekannt ist, ist es sehr einfach, für einen vorgegebenen Text x ein gültiges Paar (x, y) zu erzeugen
- Für bestimmte kryptografische Anwendungen ist es wichtig, dass dies bei vorgegebenem Hashwert y dagegen nicht möglich ist

Problem P1 (Bestimmung eines Urbilds)
Gegeben: Eine Hashfunktion $h\colon X \to Y$ und ein Hashwert $y \in Y$
Gesucht: Ein Text $x \in X$ mit $h(x) = y$

- Falls es einen immensen Aufwand erfordert, bei gegebenem Hashwert y einen Text x mit $h(x) = y$ zu finden, so heißt h ***Einweg-Hashfunktion*** *(one-way hash function bzw. preimage resistant hash function)*
- Diese Eigenschaft wird beispielsweise benötigt, wenn die Hashwerte der Benutzerpasswörter in einer öffentlich zugänglichen Datei abgespeichert werden, wie es bei manchen Unix-Systemen der Fall ist

Schlüssellose Hashfunktionen (MDCs)

- Für andere Anwendungen ist es dagegen wichtig, dass es für einen gebenen Text x praktisch unmöglich ist, einen weiteren Text $x' \neq x$ mit dem gleichen Hashwert $h(x') = h(x)$ zu finden

Problem P2 (Bestimmung eines zweiten Urbilds)
Gegeben: Eine Hashfunktion $h\colon X \to Y$ und ein Text $x \in X$
Gesucht: Ein Text $x' \in X \setminus \{x\}$ mit $h(x') = h(x)$

- Falls Problem P2 einen immensen Aufwand erfordert, heißt h ***schwach kollisionsresistent*** *(weakly collision resistant bzw. second preimage resistant)*
- Diese Eigenschaft wird beim Integritätsschutz durch einen MDC benötigt

Schlüssellose Hashfunktionen (MDCs)

- Für bestimmte Anwendungen ist es sogar nötig, dass sich überhaupt kein Kollisionspaar finden lässt
- Diese Eigenschaft ist bspw. beim Einsatz von MDCs bei der Erstellung von digitalen Signaturen erforderlich

Problem P3 (Bestimmung einer Kollision)
Gegeben: Eine Hashfunktion $h\colon X \to Y$
Gesucht: Zwei Texte $x \neq x' \in X$ mit $h(x') = h(x)$

- Falls Problem P3 einen immensen Aufwand erfordert, heißt h ***(stark) kollisionsresistent*** *(collision resistant)*

Schlüssellose Hashfunktionen (MDCs)

- Falls Problem P3 einen immensen Aufwand erfordert, heißt h ***(stark) kollisionsresistent*** *(collision resistant)*
- Obwohl die schwache Kollisionsresistenz eine gewisse Ähnlichkeit mit der Einweg-Eigenschaft aufweist, sind diese beiden Eigenschaften im allgemeinen unvergleichbar:
  - Eine schwach kollisionsresistente Funktion muss nicht notwendigerweise eine Einwegfunktion sein, da die Bestimmung eines Urbildes gerade für diejenigen Funktionswerte einfach sein kann, die nur ein einziges Urbild besitzen
  - Umgekehrt impliziert die Einweg-Eigenschaft auch nicht die schwache Kollisionsresistenz, da die Kenntnis eines Urbildes das Auffinden weiterer Urbilder sehr stark erleichtern kann

Vergleich von Sicherheitsanforderungen

- Wir zeigen nun, dass stark kollisionsresistente Hashfunktionen sowohl schwach kollisionsresistent als auch Einweghashfunktionen sind
- Hierzu reduzieren wir das Kollisionsroblem auf das Problem, ein zweites Urbild zu bestimmen

Satz

- Sei $h\colon X \to Y$ eine (n, m)-Hashfunktion
- Dann ist das Problem P3, ein Kollisionspaar für h zu bestimmen, auf das Problem P2, ein zweites Urbild zu bestimmen, reduzierbar
- Folglich sind stark kollisionsresistente Hashfunktionen auch schwach kollisionsresistent

Vergleich von Sicherheitsanforderungen

Satz

- Sei $h\colon X \to Y$ eine (n, m)-Hashfunktion
- Dann ist das Problem P3, ein Kollisionspaar für h zu bestimmen, auf das Problem P2, ein zweites Urbild zu bestimmen, reduzierbar
- Folglich sind stark kollisionsresistente Hashfunktionen auch schwach kollisionsresistent

Beweis.

- Sei A ein Las-Vegas Algorithmus, der für ein zufällig aus X gewähltes x mit Erfolgswahrscheinlichkeit ε ein zweites Urbild x' für h liefert und andernfalls ? ausgibt
- Dann ist klar, dass folgender Las-Vegas Algorithmus mit Wahrscheinlichkeit ε ein Kollisionspaar findet:

```
1 wähle zufällig x ∈ X
2 x' := A(x)
3 if x' ≠ ? then return (x, x') else return ?
```

□

Vergleich von Sicherheitsanforderungen

Als nächstes reduzieren wir das Kollisionsproblem auf das Urbildproblem

Satz

- Sei $h\colon X \to Y$ eine (n, m)-Hashfunktion mit $n \geq 2m$
- Dann ist das Problem P3, ein Kollisionspaar für h zu bestimmen, auf das Problem P1, ein Urbild zu bestimmen, reduzierbar

Beweis.

- Sei A ein Invertierungsalgorithmus für h, d.h. A berechnet für jeden Hashwert y in $W(h) = \{h(x) \mid x \in X\}$ ein Urbild x mit $h(x) = y$
- Betrachte folgenden Las-Vegas Algorithmus B:

```
1 wähle zufällig x ∈ X
2 y := h(x)
3 x' := A(y)
4 if x ≠ x' then return (x, x') else return ?
```

Vergleich von Sicherheitsanforderungen

Beweis.

- Sei A ein Invertierungsalgorithmus für h, d.h. A berechnet für jeden Hashwert y in $W(h) = \{h(x) \mid x \in X\}$ ein Urbild x mit $h(x) = y$
- Betrachte folgenden Las-Vegas Algorithmus B:

```
1 wähle zufällig x ∈ X
2 y := h(x)
3 x' := A(y)
4 if x ≠ x' then return (x, x') else return ?
```

- Sei $\mathcal{C} = \{h^{-1}(y) \mid y \in W(X)\}$
- Dann hat B eine Erfolgswahrscheinlichkeit von

$$\sum_{C \in \mathcal{C}} \frac{\|C\|}{\|X\|} \cdot \frac{\|C\| - 1}{\|C\|} = \frac{1}{n} \sum_{C \in \mathcal{C}} (\|C\| - 1) = (n - m)/n \geq \frac{1}{2}$$

□

Das Zufallsorakelmodell (ZOM)

- Das ZOM dient dazu, den Aufwand verschiedener Angriffe auf eine Hashfunktion $h\colon X \to Y$ nach oben abzuschätzen
- Sind X und Y vorgegeben, so können wir eine Hashfunktion $h\colon X \to Y$ dadurch „konstruieren“, dass wir für jedes $x \in X$ zufällig ein $y \in Y$ wählen und $h(x) = y$ setzen
- Äquivalent hierzu ist, für h eine zufällige Funktion aus der Klasse $F(X, Y)$ aller m^n Funktionen von X nach Y zu wählen
- Dieses Verfahren ist auf Grund des hohen Aufwands zwar nicht mehr praktikabel, wenn $n = \|X\|$ eine bestimmte Größe übersteigt
- Es liefert uns aber ein theoretisches Modell für eine Hashfunktion mit „idealen“ kryptografischen Eigenschaften
- Offensichtlich kann ein Angreifer nur dadurch Informationen über h erhalten, dass er für eine Reihe von Texten x_i die zugehörigen Hashwerte $h(x_i)$ berechnet (was der Befragung eines funktionalen Zufallsorakels entspricht)

Das Zufallsorakelmodell (ZOM)

Eine Zufallsfunktion h eignet sich deshalb gut als kryptografische Hashfunktion, weil der Hashwert $h(x)$ für einen Text x auch dann noch schwer vorhersagbar ist, wenn der Angreifer bereits die Hashwerte einer beliebigen Zahl von anderen Texten $x_i \neq x$ kennt

Proposition

- Sei $X_0 = \{x_1, \ldots, x_k\}$ eine beliebige Menge von k verschiedenen Texten $x_i \in X$ und seien $y_1, \ldots, y_k \in Y$
- Dann gilt für eine zufällig aus $F(X, Y)$ gewählte Funktion h und für jedes Paar $(x, y) \in (X - X_0) \times Y$,

$$\Pr[h(x) = y \,|\, h(x_i) = y_i \text{ für } i = 1, \ldots, k] = 1/m$$

Das Zufallsorakelmodell (ZOM)

- Um eine obere Komplexitätsschranke für das Urbildproblem P1 im ZOM zu erhalten, betrachten wir folgenden Algorithmus
- Hierbei gibt der Parameter q die Anzahl der Hashwertberechnungen (also die Anzahl der gestellten Orakelfragen an das Zufallsorakel h) an
- Die Laufzeit des Algorithmus ist also proportional zu q

Prozedur `FindPreimage`(h, y, q)

```
1 wähle eine beliebige Menge X_0 = {x_1, ..., x_q} ⊆ X
2 for each x_i ∈ X_0 do
3    if h(x_i) = y then return(x_i)
4 return ?
```

Das Zufallsorakelmodell (ZOM)

Satz

FindPreimage(h, y, q) gibt im ZOM mit Wahrscheinlichkeit $\varepsilon = 1 - (1 - 1/m)^q$ ein Urbild von y aus (unabhängig von der Wahl der Menge X_0)

Beweis.

- Sei $y \in Y$ fest und sei $X_0 = \{x_1, \ldots, x_q\}$
- Für $i = 1, \ldots, q$ bezeichne E_i das Ereignis "$h(x_i) = y$"
- Nach obiger Proposition sind diese Ereignisse stochastisch unabhängig und ihre Wahrscheinlichkeit ist

$$\Pr[E_i] = 1/m \text{ für } i = 1, \ldots, q$$

- Also folgt

$$\Pr[E_1 \cup \ldots \cup E_q] = 1 - \Pr[\overline{E}_1 \cap \ldots \cap \overline{E}_q] = 1 - (1 - 1/m)^q$$

□

Das Zufallsorakelmodell (ZOM)

Folgender Algorithmus liefert uns eine obere Schranke für die Komplexität des Problems P2, ein zweites Urbild für $h(x)$ zu bestimmen

Prozedur `FindSecondPreimage`(h, x, q)

```
1   y := h(x)
2   wähle eine beliebige Menge X_0 = {x_1, ..., x_{q-1}} ⊆ X − {x}
3   for each x_i ∈ X_0 do
4     if h(x_i) = y then return(x_i)
5   return ?
```

Satz

FindSecondPreimage(h, x, q) gibt im ZOM mit Wahrscheinlichkeit $\varepsilon = 1 - (1 - 1/m)^{q-1}$ ein zweites Urbild $x_0 \neq x$ von $y = h(x)$ aus.

Der Beweis ist analog zum Beweis des vorherigen Satzes

Der Geburtstagsangriff

- Ist q vergleichsweise klein, so ist bei beiden bisher betrachteten Angriffen $\varepsilon \approx q/m$
- Um also auf eine Erfolgswahrscheinlichkeit von $1/2$ zu kommen, ist $q \approx m/2$ zu wählen
- Geht es lediglich darum, *irgendein* Kollisionspaar (x, x') aufzuspüren, so bietet sich ein sogenannter ***Geburtstagsangriff*** an
- Dieser lässt sich deutlich zeiteffizienter realisieren
- Wie der Name schon andeutet, basiert dieser Angriff auf dem sog. Geburtstagsparadoxon, welches in seiner einfachsten Form folgendes besagt

Geburtstagsparadoxon

Bereits in einer Klasse mit 23 Kindern ist die Wahrscheinlichkeit größer $1/2$, dass mindestens zwei Kinder am gleichen Tag Geburtstag haben

Der Geburtstagsangriff

- Der nächste Satz besagt, dass bei q-maligem Ziehen (mit Zurücklegen) aus einer Urne mit m Kugeln mit einer Wahrscheinlichkeit von

  $$1 - (m-1)(m-2)\cdots(m-q+1)/m^{q-1}$$

  mindestens eine Kugel mehrmals gezogen wird
- Für $m = 365$ und $q = 23$ ergibt dies einen Wert von ungefähr 0,507
- Da die Häufigkeiten der Geburtstage in einer Klasse nicht gleichverteilt sind, ist die Wahrscheinlichkeit, dass 2 Kinder am gleichen Tag Geburtstag haben, sogar noch etwas höher
- Zur Kollisionsbestimmung verwenden wir folgenden Algorithmus

Prozedur `Collision`(h, q)

1 wähle eine beliebige Menge $X_0 = \{x_1, \ldots, x_q\} \subseteq X$
2 **for** each $x_i \in X_0$ **do** $y_i := h(x_i)$
3 **if** $\exists i \neq j : y_i = y_j$ **then return** (x_i, x_j) **else return** ?

Der Geburtstagsangriff

Prozedur `Collision`(h, q)

1 wähle eine beliebige Menge $X_0 = \{x_1, \ldots, x_q\} \subseteq X$
2 **for** each $x_i \in X_0$ **do** $y_i := h(x_i)$
3 **if** $\exists i \neq j : y_i = y_j$ **then return** (x_i, x_j) **else return** ?

- Bei einer naiven Implementierung würde zwar der Zeitaufwand für die Auswertung der if-Bedingung quadratisch von q abhängen
- Trägt man aber jeden Text x unter dem Suchwort $h(x)$ in eine Hashtabelle der Größe q ein, so wird der Zeitaufwand für jeden einzelnen Text x im wesentlichen durch die Berechnung von $h(x)$ bestimmt

Satz

$\textsc{Collision}(h, q)$ gibt im ZOM mit Erfolgswahrscheinlichkeit

$$\varepsilon = 1 - \frac{(m-1)(m-2)\cdots(m-q+1)}{m^{q-1}}$$

ein Kollisionspaar (x, x') für h aus

Der Geburtstagsangriff

Beweis.

- Sei $X_0 = \{x_1, \ldots, x_q\}$ und für $i = 1, \ldots, q$ bezeichne E_i das Ereignis

$$h(x_i) \notin \{h(x_1), \ldots, h(x_{i-1})\}$$

- Dann ist $E_1 \cap \ldots \cap E_q$ das Ereignis "$\textsc{Collision}(h, q)$ gibt ? aus"
- Für $i = 1, \ldots, q$ gilt nun

$$\Pr[E_i \,|\, E_1 \cap \ldots \cap E_{i-1}] = \frac{m - i + 1}{m}$$

- Dies führt auf die Erfolgswahrscheinlichkeit

$$\begin{aligned} \varepsilon &= 1 - \Pr[E_1 \cap \ldots \cap E_q] \\ &= 1 - \Pr[E_1]\Pr[E_2 \,|\, E_1] \cdots \Pr[E_q \,|\, E_1 \cap \ldots \cap E_{q-1}] \\ &= 1 - \left(\frac{m-1}{m}\right)\left(\frac{m-2}{m}\right)\cdots\left(\frac{m-q+1}{m}\right) \end{aligned}$$

□

Der Geburtstagsangriff

- Mit der Approximation $1 - x \approx e^{-x}$ erhalten wir folgende Abschätzung für ε:

$$\begin{aligned} \varepsilon &= 1 - \prod_{i=1}^{q-1} \left(1 - \frac{i}{m}\right) \\ &\approx 1 - \prod_{i=1}^{q-1} e^{\frac{-i}{m}} = 1 - e^{-\frac{1}{m}\sum_{i=1}^{q-1} i} = 1 - e^{-\frac{q(q-1)}{2m}} \\ &\approx 1 - e^{-\frac{q^2}{2m}} \approx q^2/2m \end{aligned}$$

- Für q erhalten wir daraus die Abschätzung

$$q \approx c_\varepsilon \sqrt{m}$$

  mit einer von ε abhängigen Konstante $c_\varepsilon = \sqrt{2\varepsilon}$
- Diese Abschätzung ist nur für ε-Werte nahe Null hinreichend genau

Der Geburtstagsangriff

- Aus der Abschätzung $\varepsilon \approx 1 - e^{-\frac{q^2}{2m}}$ für ε (siehe vorige Folie) erhalten wir insbesondere für größere Werte von ε eine bessere Abschätzung für q:

  $$q \approx c'_\varepsilon \sqrt{m}$$

  mit der Konstanten $c'_\varepsilon = \sqrt{2 \ln \frac{1}{1-\varepsilon}}$
- Für $\varepsilon = 1/2$ ergibt sich somit $q \approx \sqrt{(2 \ln 2)m} \approx 1{,}17\sqrt{m}$
- Besitzt also eine binäre Hashfunktion $h\colon \{0,1\}^n \to \{0,1\}^m$ die Hashwertlänge $m = 128$ Bit, so müssen im ZOM $q \approx 1{,}17 \cdot 2^{64}$ Texte gehasht werden, um mit einer Wahrscheinlichkeit von $1/2$ eine Kollision zu finden
- Um einem Geburtstagsangriff widerstehen zu können, sollte eine Hashfunktion mindestens eine Hashwertlänge von 128 oder besser 160 Bit haben

Iterierte Hashfunktionen

- Im Folgenden beschäftigen wir uns mit der Frage, wie sich aus einer kollisionsresistenten Kompressionsfunktion

  $$h\colon \{0,1\}^{m+t} \to \{0,1\}^{m}$$

  eine kollisionsresistente Hashfunktion

  $$\hat{h}\colon \{0,1\}^{*} \to \{0,1\}^{l}$$

  konstruieren lässt
- Hierzu betrachten wir folgende kanonische Konstruktionsmethode:

Iterierte Hashfunktionen

Preprocessing: Transformiere $x \in \{0,1\}^*$ mittels einer Funktion

$$y : \{0,1\}^* \to \bigcup_{r \geq 1} \{0,1\}^{rt}$$

zu einem String $y(x)$ mit der Eigenschaft $|y(x)| \equiv_t 0$

Processing: Sei $IV \in \{0,1\}^m$ ein öffentlich bekannter Initialisierungsvektor und sei $y(x) = y_1 \cdots y_r$ mit $|y_i| = t$ für $i = 1, \ldots, r$. Berechne eine Folge $z_0, \ldots, z_r$ von Strings $z_i \in \{0,1\}^m$ wie folgt:

$$z_i = \begin{cases} IV, & i = 0, \\ h(z_{i-1} y_i), & i = 1, \ldots, r \end{cases}$$

Optionale Ausgabetransformation: Berechne den Wert $\hat{h}(x) = g(z_r)$, wobei $g : \{0,1\}^m \to \{0,1\}^l$ eine öffentlich bekannte Funktion ist (meist wird für g die Identität verwendet)

Zur Berechnung von $\hat{h}(x)$ wird also die Funktion h genau r-mal aufgerufen

Iterierte Hashfunktionen

Wir formulieren nun eine für Preprocessing-Funktionen wünschenswerte Eigenschaft

Definition

- Eine Funktion $y\colon \{0,1\}^* \to \{0,1\}^*$ heißt ***suffixfrei***, falls es keine Strings $x \neq \tilde{x}$ und z in $\{0,1\}^*$ mit $y(\tilde{x}) = zy(x)$ gibt
- Mit anderen Worten: kein Funktionswert $y(x)$ ist Suffix eines Funktionswertes $y(\tilde{x})$ an einer Stelle $\tilde{x} \neq x$

Man beachte, dass jede suffixfreie Funktion insbesondere injektiv ist

Iterierte Hashfunktionen

Satz

Falls die Preprocessing-Funktion y suffixfrei und die Ausgabetransformation g injektiv ist, so ist mit h auch $\hat{h}$ kollisionsresistent

Beweis.

- Wir nehmen an, dass es gelingt, ein Kollisionspaar $(x, \tilde{x})$ für $\hat{h}$ zu finden (d.h. $\hat{h}(x) = \hat{h}(\tilde{x})$ und $x \neq \tilde{x}$)
- Seien $y(x) = y_1 \dots y_r$ und $y(\tilde{x}) = \tilde{y}_1 \dots \tilde{y}_s$ mit $r \leq s$
- Da y suffixfrei ist, muss ein Index $i \in \{1, \dots, r\}$ mit $y_i \neq \tilde{y}_{s-r+i}$ existieren
- Weiter seien z_i $(i = 0, \dots, r)$ und $\tilde{z}_j$ $(j = 0, \dots, s)$ die in der Processing-Phase berechneten Hashwerte
- Da g injektiv ist, muss mit $g(z_r) = \hat{h}(x) = \hat{h}(\tilde{x}) = g(\tilde{z}_s)$ auch $z_r = \tilde{z}_s$ gelten

Iterierte Hashfunktionen

Beweis.

- Wir nehmen an, dass es gelingt, ein Kollisionspaar $(x, \tilde{x})$ für $\hat{h}$ zu finden (d.h. $\hat{h}(x) = \hat{h}(\tilde{x})$ und $x \neq \tilde{x}$)
- Seien $y(x) = y_1 \dots y_r$ und $y(\tilde{x}) = \tilde{y}_1 \dots \tilde{y}_s$ mit $r \leq s$
- Da y suffixfrei ist, muss ein Index $i \in \{1, \dots, r\}$ mit $y_i \neq \tilde{y}_{s-r+i}$ existieren
- Weiter seien z_i $(i = 0, \dots, r)$ und $\tilde{z}_j$ $(j = 0, \dots, s)$ die in der Processing-Phase berechneten Hashwerte
- Da g injektiv ist, muss mit $g(z_r) = \hat{h}(x) = \hat{h}(\tilde{x}) = g(\tilde{z}_s)$ auch $z_r = \tilde{z}_s$ gelten
- Sei i_{max} der größte Index $i \in \{1, \dots, r\}$ mit $z_{i-1} y_i \neq \tilde{z}_{s-r+i-1} \tilde{y}_{s-r+i}$
- Dann bilden $z_{i_{max}-1} y_{i_{max}}$ und $\tilde{z}_{s-r+i_{max}-1} \tilde{y}_{s-r+i_{max}}$ wegen

$$h(z_{i_{max}-1} y_{i_{max}}) = z_{i_{max}} = \tilde{z}_{s-r+i_{max}} = h(\tilde{z}_{s-r+i_{max}-1} \tilde{y}_{s-r+i_{max}})$$

ein Kollisionspaar für h

□

Die Merkle-Damgaard-Konstruktion

- Merkle und Damgaard schlugen 1989 folgende konkrete Realisierung ihrer Konstruktion vor
- Als Initialisierungsvektor wird der Nullvektor $IV = 0^m$ benutzt, die optionale Ausgabetransformation entfällt, und für $y(x)$ wird im Fall $t \geq 2$ die folgende Funktion verwendet
  (den Fall $t = 1$ betrachten wir später)

Die Merkle-Damgaard-Konstruktion

- Für $x = \varepsilon$ sei $y(x) = 0^t$
- Für $x \in \{0,1\}^n$ mit $n > 0$ sei $r = \lceil \frac{n}{t-1} \rceil$ und $x = x_1 x_2 \ldots x_{r-1} x_r$ mit $|x_1| = |x_2| = \ldots = |x_{r-1}| = t - 1$ sowie $|x_r| = t - 1 - d$, wobei $0 \leq d < t - 1$
- Im Fall $r = 1$ ist dann $y(x) = y_1 y_2$ mit $y_1 = 0x0^d$ und $y_2 = 1bin_{t-1}(d)$
- Und für $r > 1$ ist $y(x) = y_1 \cdots y_{r+1}$, wobei

$$y_i = \begin{cases} 0x_1, & i = 1, \\ 1x_i, & 2 \leq i < r, \\ 1x_r 0^d, & i = r, \\ 1bin_{t-1}(d), & i = r+1, \end{cases} \tag{1}$$

und $bin_{t-1}(d)$ die durch führende Nullen auf die Länge $t - 1$ aufgefüllte Binärdarstellung von d ist

Die Merkle-Damgaard-Konstruktion

Satz

Die durch (1) definierte Preprocessing-Funktion y ist suffixfrei

Beweis.

- Seien $x \neq \tilde{x}$ zwei Texte mit $|x| \leq |\tilde{x}|$
- Wir müssen zeigen, dass $y(x) = y_1 y_2 \dots y_{r+1}$ kein Suffix von $y(\tilde{x}) = \tilde{y}_1 \tilde{y}_2 \dots \tilde{y}_{s+1}$ ist
- Im Fall $x = \varepsilon$ ist dies klar
- Für $x \neq \varepsilon$ machen wir folgende Fallunterscheidung

  1. Fall: $|x| \not\equiv_{t-1} |\tilde{x}|$. Dann folgt $d \neq \tilde{d}$ und somit $y_{r+1} \neq \tilde{y}_{s+1}$
  2. Fall: $|x| = |\tilde{x}|$. In diesem Fall ist $r = s$. Wegen $x \neq \tilde{x}$ existiert ein Index $i \in \{1, \dots, r\}$ mit $x_i \neq \tilde{x}_i$. Dies impliziert $y_i \neq \tilde{y}_i$, also ist $y(x)$ kein Suffix von $y(\tilde{x})$
  3. Fall: $|x| \neq |\tilde{x}|$ und $|x| \equiv_{t-1} |\tilde{x}|$. In diesem Fall ist $r < s$. Da $y(x)$ mit einer Null beginnt, aber das $(s - r + 1)$-te Bit von $y(\tilde{x})$ eine Eins ist, kann $y(x)$ kein Suffix von $y(\tilde{x})$ sein

Die Merkle-Damgaard-Konstruktion

Nun betrachten wir den Fall $t = 1$

- Sei y die durch $y(x) := 11f(x)$ definierte Funktion, wobei f wie folgt definiert ist:

$$f(x_1 \dots x_n) = f(x_1) \dots f(x_2) \text{ mit } f(0) = 0 \text{ und } f(1) = 01$$

- Dann ist leicht zu sehen, dass y suffixfrei ist □

- Da die Kompressionsfunktion h bei der Berechnung von $\hat{h}(x)$ im Fall $t = 1$ für jedes Bit von $y(x)$ einmal aufgerufen wird, wird h genau $|y(x)| \leq 2(n+1)$-mal aufgerufen
- Im Fall $t > 1$ werden dagegen nur $r + 1 = \lceil \frac{n}{t-1} \rceil + 1$ Aufrufe benötigt

Die MD4-Hashfunktion

- Die MD4-Hashfunktion (***M**essage **D**igest*) wurde 1990 von Rivest vorgeschlagen
- Die Bitlänge von MD4 beträgt $l = 128$ Bit
- Bei einer Wortlänge von 32 Bit entspricht dies 4 Wörtern
- MD4 und die im Folgenden vorgestellten Hashfunktionen benutzen u.a. folgende Operationen auf Wörtern $X, Y \in \{0,1\}^{32}$

Wort-Operationen	
$X \wedge Y$	bitweises „Und“ von X und Y
$X \vee Y$	bitweises „Oder“ von X und Y
$X \oplus Y$	bitweises „exklusives Oder“ von X und Y
$\neg X$	bitweises Komplement von X
$X + Y$	Ganzzahl-Addition modulo 2^{32}
$X \rightarrow s$	Rechtsshift um s Stellen
$X \hookleftarrow s$	zirkulärer Linksshift um s Stellen

Die MD4-Hashfunktion

- Die Ganzzahl-Addition wird bei MD4 und MD5 in little endian Architektur ausgeführt
- D.h. dass ein aus 4 Bytes, zusammengesetztes Wort $X = a_3 a_2 a_1 a_0$, dessen Bytes $a_i \in 2^8$ die Zahlenwerte $(a_i)_2 \in [0, 255]$ haben, die Zahl $(a_0)_2 2^{24} + (a_1)_2 2^{16} + (a_2)_2 2^8 + (a_3)_2$ repräsentiert
- Dagegen verwendet `SHA-1` eine big endian Architektur
- D.h. dass $X = a_3 a_2 a_1 a_0$ die Zahl $(a_3)_2 2^{24} + (a_2)_2 2^{16} + (a_1)_2 2^8 + (a_0)_2$ repräsentiert
- Der MD4-Algorithmus benutzt die folgenden Funktionen f_j für $j = 0, \ldots, 47$:

$$f_j(X, Y, Z) := \begin{cases} (X \wedge Y) \vee (\neg X \wedge Z), & j = 0, \ldots, 15 \\ (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z), & j = 16, \ldots, 31 \\ X \oplus Y \oplus Z, & j = 32, \ldots, 47 \end{cases}$$

Die MD4-Hashfunktion

- Zudem benutzt er die folgenden Konstanten y_j, z_j, s_j für $j = 0, \ldots, 47$:

	y_j (in Hexadezimaldarstellung)
$j = 0, \ldots, 15$	0
$j = 16, \ldots, 31$	5*a*827999
$j = 32, \ldots, 47$	6*ed*9*eba*1

	z_j
$j = 0, \ldots, 15$	$0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15$
$j = 16, \ldots, 31$	$0, 4, 8, 12, 1, 5, 9, 13, 2, 6, 10, 14, 3, 7, 11, 15$
$j = 32, \ldots, 47$	$0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15$

	s_j
$j = 0, \ldots, 15$	$3, 7, 11, 19, 3, 7, 11, 19, 3, 7, 11, 19, 3, 7, 11, 19$
$j = 16, \ldots, 31$	$3, 5, 9, 13, 3, 5, 9, 13, 3, 5, 9, 13, 3, 5, 9, 13$
$j = 32, \ldots, 47$	$3, 9, 11, 15, 3, 9, 11, 15, 3, 9, 11, 15, 3, 9, 11, 15$

Die MD4-Hashfunktion

MD4(x)

1 **input** $x \in \{0,1\}^*, |x| = n$
2 $y := x10^k bin_{64}(n)$, $k \in \{0,1,\ldots,511\}$ mit
$n + 1 + k + 64 \equiv 0 \pmod{512}$
3 $(H_1, H_2, H_3, H_4) := (67452301, efcdab89, 98badcfe, 10325476)$
4 sei $y = M_1 \cdots M_r$, $r = (n + 1 + k + 64)/512$
5 **for** $i := 1$ **to** r **do**
6 sei $M_i = X[0] \cdots X[15]$
7 $(A, B, C, D) := (H_1, H_2, H_3, H_4)$
8 **for** $j := 0$ **to** 47 **do**
9 $(A, B, C, D) := (D, (A + f_j(B, C, D) + X[z_j] + y_j) \hookleftarrow s_j, B, C)$
10 $(H_1, H_2, H_3, H_4) := (H_1 + A, H_2 + B, H_3 + C, H_4 + D)$
11 **output** $H_1 H_2 H_3 H_4$

- In Zeile 9 wird die Kompressionsfunktion von MD4 berechnet:
$(A, B, C, D, X[z_j]) \mapsto (D, (A + f_j(B, C, D) + X[z_j] + y_j) \hookleftarrow s_j, B, C)$

Die MD4-Hashfunktion

- Für MD4 konnten nach ca. 2^{20} Hashwertberechnungen Kollisionen aufgespürt werden
- Deshalb gilt MD4 heutzutage nicht mehr als kollisionsresistent

Die MD5-Hashfunktion

- Der MD5 ist eine 1991 von Rivest präsentierte verbesserte Version von MD4
- Die Bitlänge von MD5 beträgt wie bei MD4 $l = 128$ Bit
- Bei einer Wortlänge von 32 Bit entspricht dies 4 Wörtern
- In MD5 werden teilweise andere Konstanten als in MD4 verwendet
- Zudem besitzt MD5 eine zusätzliche 4. Runde ($j = 48, \ldots, 63$), in der die Funktion $f_j(X, Y, Z) = Y \oplus (X \vee \neg Z)$ verwendet wird
- Außerdem wurde die in Runde 2 von MD4 verwendete Funktion durch $f_j(X, Y, Z) := (X \wedge Z) \vee (Y \wedge \neg Z)$, $j = 16 \ldots 31$, ersetzt
- Die y-Konstanten sind definiert als

  $y_j :=$ die ersten 32 Bit der Binärdarstellung von $\text{abs}(\sin(j + 1))$, $0 \leq j \leq 63$,

Die MD5-Hashfunktion

- Zudem benutzt der MD5 die folgenden Konstanten z_j und s_j:

j	z_j
$0, \ldots, 15$	$z_j = j$: $0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15$
$16, \ldots, 31$	$z_j = (5j + 1) \bmod 16$: $1, 6, 11, 0, 5, 10, 15, 4, 9, 14, 3, 8, 13, 2, 7, 12$
$32, \ldots, 47$	$z_j = (3j + 5) \bmod 16$: $5, 8, 11, 14, 1, 4, 7, 10, 13, 0, 3, 6, 9, 12, 15, 2$
$48, \ldots, 63$	$z_j = 7j \bmod 16$: $0, 7, 14, 5, 12, 3, 10, 1, 8, 15, 6, 13, 4, 11, 2, 9$

j	s_j
$0, \ldots, 15$	$7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22$
$16, \ldots, 31$	$5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 14, 20$
$32, \ldots, 47$	$4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23$
$48, \ldots, 63$	$6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21$

Die MD5-Hashfunktion

MD5(x)

input $x \in \{0,1\}^*, |x| = n$
$y := x10^k bin_{64}(n)$, $k \in \{0,1,\ldots,511\}$ mit
$n + 1 + k + 64 \equiv 0 \pmod{512}$
$(H_1, H_2, H_3, H_4) := (67452301, efcdab89, 98badcfe, 10325476)$
sei $y = M_1 \cdots M_r$, $r = (n + 1 + k + 64)/512$
for $i := 1$ **to** r **do**
  sei $M_i = X[0] \cdots X[15]$
  $(A, B, C, D) := (H_1, H_2, H_3, H_4)$
  for $j := 0$ **to** 63 **do**
    $(A, B, C, D) := (D, B + (A + f_j(B, C, D) + X[z_j] + y_j) \hookleftarrow s_j, B, C)$
  $(H_1, H_2, H_3, H_4) := (H_1 + A, H_2 + B, H_3 + C, H_4 + D)$
output $H_1 H_2 H_3 H_4$

- Für MD5 konnten in 2004 ebenfalls Kollisionspaare gefunden werden
- Für die Kompressionsfunktion von MD5 gelang dies bereits 1996

Die SHA-1-Hashfunktion

- Der Secure Hash Algorithm (SHA-1) ist eine Weiterentwicklung des MD4 bzw. MD5 Algorithmus
- Er gilt in den USA als Standard und ist Bestandteil des von der US-Behörde NIST (National Institute of Standards and Technology) im August 1991 veröffentlichten DSS (Digital Signature Standard)
- Die Bitlänge von SHA-1 beträgt $l = 160$ Bit
- Bei einer Wortlänge von 32 Bit entspricht dies 5 Wörtern
- SHA-1 unterscheidet sich nur geringfügig von der SHA-0 Hashfunktion, in der eine Schwachstelle dazu führt, dass nach Berechnung von ca. 2^{61} Hashwerten ein Kollisionspaar gefunden werden kann (obwohl bei einem Geburtstagsangriff auf Grund der Hashwertlänge von 160 Bit ca. 2^{80} Berechnungen erforderlich sein müssten)
- Diese potentielle Schwäche von SHA-0 wurde im SHA-1 dadurch entfernt, dass SHA-1 in Zeile 8 einen zirkulären Shift um eine Bitstelle ausführt

Die SHA-1-Hashfunktion

- Der SHA-1-Algorithmus benutzt die folgenden Konstanten K_j für $j = 0, \ldots, 79$:

	K_j (in Hexadezimaldarstellung)
$j = 0, \ldots, 19$	$5a827999$
$j = 20, \ldots, 39$	$6ed9eba1$
$j = 40, \ldots, 59$	$8f1bbcdc$
$j = 60, \ldots, 79$	$ca62c1d6$

und folgende Funktionen f_j für $j = 0, \ldots, 79$:

$$f_j(X, Y, Z) := \begin{cases} (X \wedge Y) \vee (\neg X \wedge Z), & j = 0, \ldots, 19 \\ X \oplus Y \oplus Z, & j = 20, \ldots, 39 \\ (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z), & j = 40, \ldots, 59 \\ X \oplus Y \oplus Z, & j = 60, \ldots, 79 \end{cases}$$

Die SHA-1-Hashfunktion

SHA-1(x)

1 **input** $x \in \{0,1\}^*, |x| = n$
2 $y := x10^k bin_{64}(n)$, $k \in \{0,1,\ldots,511\}$ mit
$n + 1 + k + 64 \equiv 0 \pmod{512}$
3 $(H_0,\ldots,H_4) := (67452301, efcdab89, 98badcfe, 10325476, c3d2e1f0)$
4 sei $y = M_1 \cdots M_r$, $r = (n+1+k+64)/512$
5 **for** $i := 1$ **to** r **do**
6 sei $M_i = X[0] \cdots X[15]$
7 **for** $t := 16$ **to** 79 **do**
8 $X[t] := (X[t-3] \oplus X[t-8] \oplus X[t-14] \oplus X[t-16]) \hookleftarrow 1$
9 $(A,B,C,D,E) := (H_0,H_1,H_2,H_3,H_4)$
10 **for** $j := 0$ **to** 79 **do**
11 $temp := (A \hookleftarrow 5) + f_j(B,C,D) + E + X[j] + K_j$
12 $(A,B,C,D,E) := (temp, A, B \hookleftarrow 30, C, D)$
13 $(H_0,\ldots,H_4) := (H_0 + A,\ldots,H_4 + E)$
14 **output** $H_0H_1H_2H_3H_4$

Die SHA-2-Familie

- Im Jahr 2001 veröffentlichte die US-Behörde NIST drei weitere Hashfunktionen der SHA-Familie: SHA-256, SHA-384, and SHA-512
- Diese Funktionen werden auch als SHA-2 Hashfunktionen bezeichnet
- In 2004 kam noch SHA-224 als vierte Variante hinzu
- SHA-256 und SHA-512 haben denselben Aufbau, unterscheiden sich aber in erster Linie in der benutzten Wortlänge: 32 Bit bei SHA-256 und 64 Bit bei SHA-512
- Zudem werden unterschiedliche Shift- und Summationskonstanten verwendet und auch die Rundenzahlen differieren
- SHA-224 und SHA-384 sind reduzierte Varianten von SHA-256 und SHA-512

Die SHA-2-Familie

- Der SHA-256-Algorithmus benutzt die folgenden Konstanten K_j, $j = 0, \ldots, 63$ (in Hexadezimaldarstellung):

  428a2f98, 71374491, b5c0fbcf, e9b5dba5, 3956c25b, 59f111f1, 923f82a4, ab1c5ed5,
  d807aa98, 12835b01, 243185be, 550c7dc3, 72be5d74, 80deb1fe, 9bdc06a7, c19bf174,
  e49b69c1, efbe4786, 0fc19dc6, 240ca1cc, 2de92c6f, 4a7484aa, 5cb0a9dc, 76f988da,
  983e5152, a831c66d, b00327c8, bf597fc7, c6e00bf3, d5a79147, 06ca6351, 14292967,
  27b70a85, 2e1b2138, 4d2c6dfc, 53380d13, 650a7354, 766a0abb, 81c2c92e, 92722c85,
  a2bfe8a1, a81a664b, c24b8b70, c76c51a3, d192e819, d6990624, f40e3585, 106aa070,
  19a4c116, 1e376c08, 2748774c, 34b0bcb5, 391c0cb3, 4ed8aa4a, 5b9cca4f, 682e6ff3,
  748f82ee, 78a5636f, 84c87814, 8cc70208, 90befffa, a4506ceb, bef9a3f7, c67178f2

- Dies sind jeweils die ersten 32 Bit der binären Nachkommastellen der dritten Wurzeln der ersten 64 Primzahlen $2, \ldots, 311$

Die SHA-256-Hashfunktion

1 **input** $x \in \{0,1\}^*, |x| = n$
2 $y := x10^k bin_{64}(n)$, $k \in \{0,\ldots,511\}$ mit $n+1+k+64 \equiv 0 \pmod{512}$
3 $(H_0,\ldots,H_7) := (6a09e667,\ldots,5be0cd19)$
4 sei $y = M_1 \cdots M_r$, $r = (n+1+k+64)/512$
5 **for** $i := 1$ **to** r **do**
6 sei $M_i = X[0] \cdots X[15]$
7 **for** $t := 16$ **to** 63 **do**
8 $s0 := (X[t-15] \hookrightarrow 7) \oplus (X[t-15] \hookrightarrow 18) \oplus (X[t-15] \to 3)$
9 $s1 := (X[t-2] \hookrightarrow 17) \oplus (X[t-2] \hookrightarrow 19) \oplus (X[t-2] \to 10)$
10 $X[t] := X[t-16] + s0 + X[t-7] + s1$
11 $(A,B,C,D,E,F,G,H) := (H_0,H_1,H_2,H_3,H_4,H_5,H_6,H_7)$
12 **for** $j := 0$ **to** 63 **do** α
13 $(H_0,H_1,\ldots,H_7) := (H_0+A, H_1+B,\ldots,H_6+G,H_7+H)$
14 **output** $H_0H_1H_2H_3H_4H_5H_6H_7$

Die Werte von $H_0,\ldots,H_7$ in Zeile 3 sind die ersten 32 Bit der binären Nachkommastellen der Wurzeln der Primzahlen $2,3,5,7,11,13,17,19$

Die SHA-256-Hashfunktion

Programmstück α

1 $s0 := (A \hookrightarrow 2) \oplus (A \hookrightarrow 13) \oplus (A \hookrightarrow 22)$
2 $maj := (A \wedge B) \oplus (A \wedge C) \oplus (B \wedge C)$
3 $t2 := s0 + maj$
4 $s1 := (E \hookrightarrow 6) \oplus (E \hookrightarrow 11) \oplus (E \hookrightarrow 25)$
5 $ch := (E \wedge F) \oplus (\neg E \wedge G)$
6 $t1 := H + s1 + ch + K_j + X[j]$
7 $(A, B, C, D, E, F, G, H) := (t1 + t2, A, B, C, D + t1, E, F, G)$

Kryptoanalyse von Hashfunktionen (MD4 und MD5)

- Bereits 1991 wurden von den Boer und Bosselaers Schwächen im MD4 aufgedeckt
- Im August 2004 erschien ein Bericht [1] mit einer Anleitung, wie sich Kollisionen für MD4 mittels “hand calculation” finden lassen
- In 1993, fanden den Boer und Bosselaers einen Weg, so genannte “Pseudo-Kollisionen” für die MD5 Kompressionsfunktion zu generieren
- In 1996, fand Dobbertin ein Kollisionspaar für die MD5 Kompressionsfunktion
- Im August 2004 wurden schließlich Kollisionen für MD5 von Xiaoyun Wang, Dengguo Feng, Xuejia Lai und Hongbo Yu berechnet
- Der benötigte Aufwand wurde mit ca. 1 Stunde auf einem IBM p690 Cluster abgeschätzt

Kryptoanalyse von Hashfunktionen (MD5)

- Im März 2005 veröffentlichten Arjen Lenstra, Xiaoyun Wang und Benne de Weger zwei X.509 Zertifikate mit unterschiedlichen Public-keys, die auf denselben MD5-Hashwert führten
- Nur wenige Tage später beschrieb Vlastimil Klima eine Möglichkeit, Kollisionen für MD5 innerhalb weniger Stunden auf einem Notebook zu berechnen
- Mittels der so genannten Tunneling-Methode wurde die Rechenzeit vom gleichen Autor im März 2006 auf eine Minute verkürzt
- Auf der CRYPTO 98 stellten Chabaud und Joux einen Angriff auf SHA-0 vor, der ein Kollisionspaar mit nur 2^{61} Hashwertberechnungen (anstelle von 2^{80} bei einem Geburtstagsangriff) aufspürt

Kryptoanalyse von Hashfunktionen (SHA-0)

- In 2004 fanden Biham und Chen Beinahe-Kollisionen für den SHA-0, bei denen sich die Hashwerte nur an 18 von den 160 Bitpositionen unterschieden
- Zudem legten sie volle Kollisionen für den auf 62 Runden reduzierten SHA-0 Algorithmus vor
- Schließlich wurde im August 2004 die Berechnung einer Kollision für den vollen 80-Runden SHA-0 Algorithmus von Joux, Carribault, Lemuet und Jalby bekannt gegeben
- Hierzu wurden lediglich 2^{51} Hashwerte berechnet, die ca. 80 000 Stunden CPU-Rechenzeit auf einem 2-Prozessor 256-Itanium Supercomputer benötigten

Kryptoanalyse von Hashfunktionen (SHA-0)

- Ebenfalls im August 2004 wurde von Wang, Feng, Lai und Yu auf der CRYPTO 2004 eine Angriffsmethode für MD5, SHA-0 und andere Hashfunktionen vorgestellt, mit der sich die Anzahl der Hashwertberechnungen auf 2^{40} senken lässt
- Dies wurde im Februar 2005 von Xiaoyun Wang, Yiqun Lisa Yin und Hongbo Yu geringfügig auf 2^{39} Hashwertberechnungen verbessert
- Aufgrund der erfolgreichen Angriffe auf SHA-0 rieten mehrere Experten von einer weiteren Verwendung des `SHA-1` ab. Daraufhin kündigte die amerikanische Behörde NIST an, `SHA-1` in 2010 zugunsten der SHA-2 Varianten abzulösen

Kryptoanalyse von Hashfunktionen (SHA-1 und SHA-2)

- Im Jahr 2005 veröffentlichten Rijmen und Oswald einen Angriff, der mit weniger als 2^{80} Hashwertberechnungen ein Kollisionspaar für den auf 53 Runden reduzierten `SHA-1` Algorithmus findet
- Nur wenig später kündigten Xiaoyun Wang, Yiqun Lisa Yin und Hongbo Yu einen Angriff auf den vollen 80-Runden `SHA-1` mit 2^{69} Hashwertberechnungen an
- Im August 2005 erfuhr der benötigte Aufwand von Xiaoyun Wang, Andrew Yao und Frances Yao auf der CRYPTO 2005 eine weitere Reduktion auf 2^{63} Berechnungen
- In 2008 wurde von Stephane Manuel ein Kollisionsangriff mit einem geschätzten Aufwand von 2^{51} bis 2^{57} Berechnungen veröffentlicht
- Im Februar 2017 fanden Stevens, Bursztein, Karpman, Albertini und Markov die erste Kollision für SHA-1
- Die besten bekannten Angriffe gegen SHA-2 brechen die von 64 auf 41 Runden reduzierte Variante von SHA-256 und die von 80 auf 46 Runden reduzierte Variante von SHA-512

Der SHA-3 Algorithmus

- Im Oktober 2012 wurde der Hash-Algorithmus Keccak als Gewinner des vom NIST ausgeschriebenen Wettbewerbs für den SHA-3-Algorithmus ausgewählt
- Die Intention dabei war nicht, SHA-2 als Standard durch SHA-3 abzulösen, zumal bisher keine erfolgreichen Angriffe gegen SHA-2 bekannt sind
- Vielmehr ging es bei diesem Wettbewerb darum, angesichts der erfolgreichen Angriffe gegen MD5 und SHA-0, die einen ähnlichen Aufbau wie SHA-1 und SHA-2 haben, eine auf einem vollkommen anderen Entwurfsprinzip basierende Alternative zur Verfügung zu stellen

Die Sponge-Konstruktion

- Die Konstruktionsidee hinter dem SHA-3-Gewinner Keccak wird von den Autoren als *Sponge* (Schwamm) bezeichnet
- Auf der Basis dieser Entwurfsmethode lassen sich außer Hashfunktionen bspw. auch Pseudozufallsgeneratoren gewinnen
- Der Aufbau eines Sponges ähnelt oberflächlich betrachtet der bereits vorgestellten Konstruktion von iterierten Hashfunktionen, weist aber einige Unterschiede auf
- So basiert ein Sponge statt auf einer Kompressionsfunktion h auf einer Permutation (oder allgemeiner Transformation) $f : \{0,1\}^b \to \{0,1\}^b$, die wie h iteriert angewendet wird
- Dabei wird der aktuelle b-Bitblock in zwei Teilblöcke der Länge r und c unterteilt, die als äußerer bzw. innerer Zustand bezeichnet werden

Die Sponge-Konstruktion

- Wie der Name schon sagt, verbleiben die Bits des inneren Zustands im Sponge, d.h. sie dienen nur zur Berechnung des nächsten Zustands und werden im Gegensatz zu den Bits des äußeren Zustands nicht unmittelbar für die Gewinnung der Ausgabe genutzt
- Die Anzahl c der Bits des inneren Zustands wird als ***Kapazität*** des Sponges bezeichnet und ist sein wichtigster Sicherheitsparameter
- Die Anzahl r der Bits des äußeren Zustands heißt ***Bitrate***, wobei $r + c = b$ gelten muss

Die Sponge-Konstruktion

- Bevor die Funktion f im Kern des Algorithmus iteriert angewendet wird, um eine Zustandsfolge zu generieren, wird ein Preprocessing ausgeführt
- Die Anforderungen an diese Funktion beschreiben wir vorab

Definition

- Eine Funktion $y\colon \{0,1\}^* \to \bigcup_{k\geq 1}\{0,1\}^{kr}$ der Form $y(x) = xz$ heißt ***Paddingfunktion*** für Bitrate $r \geq 1$
- Eine solche Funktion heißt ***sponge-konform*** für Bitrate $r \geq 1$, falls gilt:
  - $\forall n \geq 0\, \exists z \forall x \in \{0,1\}^n : y(x) = xz$
  - $\forall k \geq 0\, \forall x \neq x' : y(x) \neq y(x')0^{kr}$

Die Sponge-Konstruktion

Beispiel

- Es ist leicht zu sehen, dass die Funktion

$$\texttt{pad10*1}_r(x) = x10^d1 \text{ mit } d = \min\{i \geq 0 \mid i + 2 + |x| \equiv_r 0\}$$

  eine sponge-konforme Paddingfunktion für die Bitrate r ist
- Tatsächlich ist $\texttt{pad10*1}_r$ sogar für jede Bitrate $r' \geq 1$ sponge-konform
- Ohne die 1 am Ende von $\texttt{pad10*1}_r(x) = x10^d1$ wäre dies nicht der Fall

Die Sponge-Konstruktion

Definition

- Sei y eine Paddingfunktion für $r \geq 1$ und sei $f : \{0,1\}^b \to \{0,1\}^b$
- Für $x \in \{0,1\}^*$ sei $y(x) = y_1 \dots y_k$ mit $|y_i| = r$ für $i = 1, \dots, k$
- Wir definieren die Zustände

$$s_i = \begin{cases} 0^b & i = 0 \\ f(s_{i-1} \oplus (y_i 0^c)) & 1 \leq i \leq k \quad \text{(Absorptionsphase)} \\ f(s_{i-1}) & i > k \quad \text{(Squeezing-Phase)} \end{cases}$$

- Weiter bezeichne z_i für $i \geq 1$ die ersten r Bit von s_{k+i-1}
- Zudem sei $m = \lfloor \frac{l}{r} \rfloor$ und z'_{m+1} sei das Präfix von z_{m+1} der Länge $l - mr$
- Dann ist die Funktion $\texttt{Sponge}_{f,y,r} : \mathbb{N} \times \{0,1\}^* \to \{0,1\}^*$ wie folgt definiert: $\texttt{Sponge}_{f,y,r}(l, x) = z_1 \dots z_m z'_{m+1}$
- Für die Analyse definieren wir noch die Funktionen

$$\texttt{Absorb}_{f,r}(y_1 \dots y_k) = s_k \text{ und } \texttt{Squeeze}_{f,r}(l, s_k) = z_1 \dots z_m z'_{m+1}$$

Die Sponge-Konstruktion

- Den Aufwand, für festes l ein Kollisionspaar x, x' mit $x \neq x'$ und $\mathtt{Sponge}_{f,y,r}(l,x) = \mathtt{Sponge}_{f,y,r}(l,x')$ zu finden, können wir nach oben durch den Aufwand abschätzen, ein Paar $x, x' \in \bigcup_{k\geq 1}\{0,1\}^{kr}$ mit $x \neq x'$ und $\mathtt{Absorb}_{f,r}(y(x)) = \mathtt{Absorb}_{f,r}(y(x'))$ zu finden
- Hierbei reicht es, ein inneres Kollisionspaar, d.h. zwei Strings $w = y_1 \ldots, y_k \neq w' = y'_1 \ldots, y'_{k'}$ zu finden, so dass die inneren Zustände von $s_k = \mathtt{Absorb}_{f,r}(w)$ und $s'_{k'} = \mathtt{Absorb}_{f,r}(w')$ gleich sind
- Setzen wir nämlich y_{k+1} und $y_{k'+1}$ auf die äußeren Zustände von s_k und $s'_{k'}$, so folgt für die Eingaben $x = wy_{k+1}$ und $x' = w'y'_{k'+1}$:

$$\begin{aligned}\mathtt{Absorb}_{f,r}(x) &= f(s_k \oplus (y_{k+1}0^c)) = f(0^r s_k^{\mathrm{i}}) = f(0^r s_{k'}'^{\mathrm{i}}) \\ &= f(s'_{k'} \oplus (y'_{k'+1}0^c)) = \mathtt{Absorb}_{f,r}(x')\end{aligned}$$

  wobei s_j^{i} den inneren Zustand von s_j bezeichnet
- Falls das Suffix z von $y(x) = xz$ nur von $|x|$ mod r abhängt, gilt wegen $|x| \equiv_r |x'|$ dann auch die Gleichheit $\mathtt{Absorb}_{f,r}(y(x)) = \mathtt{Absorb}_{f,r}(y(x'))$ und somit $\mathtt{Sponge}_{f,y,r}(l,x) = \mathtt{Sponge}_{f,y,r}(l,x')$

Die Sponge-Konstruktion

- Um eine solche innere Kollision zu finden, hilft es, sich die 2^c inneren Zustände $u \in \{0,1\}^c$ als Knoten eines gerichteten Multigraphen G vorzustellen, der für jedes Paar $(xu, x'u')$ mit $f(xu) = x'u'$ eine Kante $(u, u')_{x,x'}$ von u nach u' mit dem Label x, x' enthält
- Ziel ist es dann, zwei verschiedene Pfade von 0^c zu demselben Knoten v zu finden, wobei zwei Pfade auch dann verschieden sind, wenn sich die Kanten nur in den Labeln unterscheiden
- Wird f durch eine Zufallsfunktion modelliert (ZOM), so lassen bereits berechnete Werte von f keine Rückschlüsse auf die Werte für andere Argumente zu
- Anders als beim ZOM für eine Hashfunktion kann es sich dennoch für den Angreifer lohnen, die Argumente von f adaptiv nach einer Strategie $\mathcal{S}$ zu wählen
- Der Algorithmus `InnerCollision` fasst dieses Vorgehen zusammen

Bestimmung eines inneren Kollisionspaares

Prozedur `InnerCollision`$(f, r, q, \mathcal{S})$

1 $c := b - r$, wobei $f : \{0,1\}^b \to \{0,1\}^b$
2 initialisiere den Multi-Digraphen $G = (V, A) := (\{0,1\}^c, \emptyset)$
3 **for** $i := 1$ **to** q **do**
4 wähle $u \in V$ und $x \in \{0,1\}^r$ nach Strategie $\mathcal{S}$
5 $x'u' := f(xu)$
6 $A := A \cup \{(u, u')_{x,x'}\}$
7 **if** $\exists$ zwei Pfade $(0^c, u_1)_{x_0,x_0'}, (u_1, u_2)_{x_1,x_1'}, \ldots, (u_{k-1}, u_k)_{x_{k-1},x_{k-1}'}$ und
8 $(0^c, v_1)_{y_0,y_0'}, (v_1, v_2)_{y_1,y_1'}, \ldots, (v_{l-1}, v_l)_{y_{l-1},y_{l-1}'}$ in G mit $u_k = v_l$ **then**
9 **return**$(x_0(x_0' \oplus x_1)\ldots(x_{k-2}' \oplus x_{k-1}), y_0(y_0' \oplus y_1)\ldots(y_{l-2}' \oplus y_{l-1}))$
10 **else**
11 **return**$(?)$

Die Sponge-Konstruktion

Satz

- Für jede Strategie $\mathcal{S}$ gibt $\textsc{InnerCollision}(f, r, q, \mathcal{S})$ im ZOM mit Erfolgswahrscheinlichkeit höchstens

$$\varepsilon = 1 - \prod_{i=1}^{q} \left(1 - \frac{i}{2^c}\right)$$

  ein inneres Kollisionspaar (x, x') aus
- Wählt $\mathcal{S}$ nur von 0^c aus erreichbare Knoten u und kein Argument xu mehrmals, so ist die Erfolgswahrscheinlichkeit exakt ε

Die Sponge-Konstruktion

Beweis.

- Sei E_i das Ereignis "G enthält nach dem i-ten Durchlauf noch keine zwei verschiedenen Pfade von 0^c zu einem Knoten v"
- Da nur durch eine Kante zwischen zwei von 0^c aus erreichbaren Knoten ein zweiter Pfad von 0^c aus geschlossen werden kann und nach $i-1$ Durchläufen höchstens i von 2^c Knoten erreichbar sind, gilt

$$\Pr[E_i \mid E_1 \cap \ldots \cap E_{i-1}] \geq 1 - \frac{i}{2^c}$$

- Wählt $\mathcal{S}$ nur erreichbare Knoten u und kein Argument xu mehrfach, so sind unter Annahme von $E_1 \cap \ldots \cap E_{i-1}$ auch i Knoten erreichbar (sonst gäbe es bereits zwei Pfade von 0^c zu einem Knoten in G) und es gilt sogar Gleichheit
- Dies führt auf eine Erfolgswahrscheinlichkeit von

$$\begin{aligned} 1-\Pr[E_1 \cap \ldots \cap E_q] &= 1-\Pr[E_1]\Pr[E_2 \mid E_1] \cdots \Pr[E_q \mid E_1 \cap \ldots \cap E_{q-1}] \\ &\leq 1 - \left(1-\frac{1}{2^c}\right)\left(1-\frac{2}{2^c}\right)\cdots\left(1-\frac{q}{2^c}\right) = \varepsilon \end{aligned}$$

□

Die Sponge-Konstruktion

- Mit der Approximation $1 - x \approx e^{-x}$ erhalten wir die Abschätzung

$$\begin{aligned}\varepsilon &= 1 - \prod_{i=1}^{q}\left(1 - \frac{i}{2^c}\right) \approx 1 - \prod_{i=1}^{q} e^{\frac{-i}{2^c}} = 1 - e^{-\frac{1}{2^c}\sum_{i=1}^{q} i} \\ &= 1 - e^{-\frac{q(q+1)}{2\cdot 2^c}} \approx 1 - e^{-\frac{q^2}{2\cdot 2^c}} \approx q^2/2\cdot 2^c\end{aligned}$$

- Für q ergibt sich daraus die Abschätzung

$$q \approx c_\varepsilon\sqrt{2^c}$$

mit einer von ε abhängigen Konstanten $c_\varepsilon = \sqrt{2\varepsilon}$

SHA-3

- Der Standard SHA-3 definiert die oben beschriebene Sponge-Konstruktion, 7 verschiedene bijektive Funktionen $f_w, w = 2^i, i \in \{0, \ldots, 6\}$ als Kern von $\mathrm{Sponge}_{f_w, \mathtt{pad10^*1}_r, r}$, sowie verschiedene Kombinationen von Bitraten r und Ausgabelängen l (c ist durch $25w - r$ bestimmt)
- Jede Funktion $f_w : \{0,1\}^{5\times5\times w} \to \{0,1\}^{5\times5\times w}$ bildet ein zweidimensionales Feld A aus w-Bit-Wörtern auf ein ebensolches Feld $f_w(A)$ ab
- Dabei wird $(12 + \log_2 w)$-mal eine Rundenfunktion $f'_w : \{0,1\}^{5\times5\times w} \times \{0,1\}^w \to \{0,1\}^{5\times5\times w}$ aufgerufen, die A und eine Rundenkonstante RC_i auf A' abbildet
- Es gilt

$$f'_w(A, RC) = \iota_{RC}(\chi(\pi(\rho(\theta(A))))),$$

wobei θ, ρ, π, χ und ι_{RC} Bijektionen von $\{0,1\}^{5\times5\times w}$ nach $\{0,1\}^{5\times5\times w}$ sind

SHA-3

- Die Funktion θ besteht aus $\oplus$-Operationen und ist so gewählt, dass sich $\theta^{-1}(A)$ an möglichst vielen Bits ändert, falls eines in A geflippt wird
- Danach permutieren die Funktionen ρ und π die Bits von A innerhalb und zwischen den Wörtern
- Ähnlich einer S-Box im SPN ist χ eine nichtlineare Funktion (die einzige solche in der Definition von f'_w), die nur auf 5-Bit-Blöcken arbeitet (jedes Bit hängt sogar nur von 2 anderen ab)
- Schlussendlich setzt ι_{RC} das Wort $A_{0,0}$ auf $A_{0,0} \oplus RC$
- Für die Werte $l \in \{224, 256, 384, 512\}$ definiert der Standard FIPS (Federal Information Processing Standard) 202:

$$\texttt{SHA3-}l(x) = \texttt{Sponge}_{f_{64},\texttt{pad10*1}_r,r}(l, x01), \qquad \text{wobei } r = 1600 - 2l$$

- Das zusätzliche Padding 01 soll dabei `SHA-3` von anderen Anwendungen von Keccak mit denselben Werten w, l, r unterscheiden

Nachrichten-Authentikationscodes (MACs)

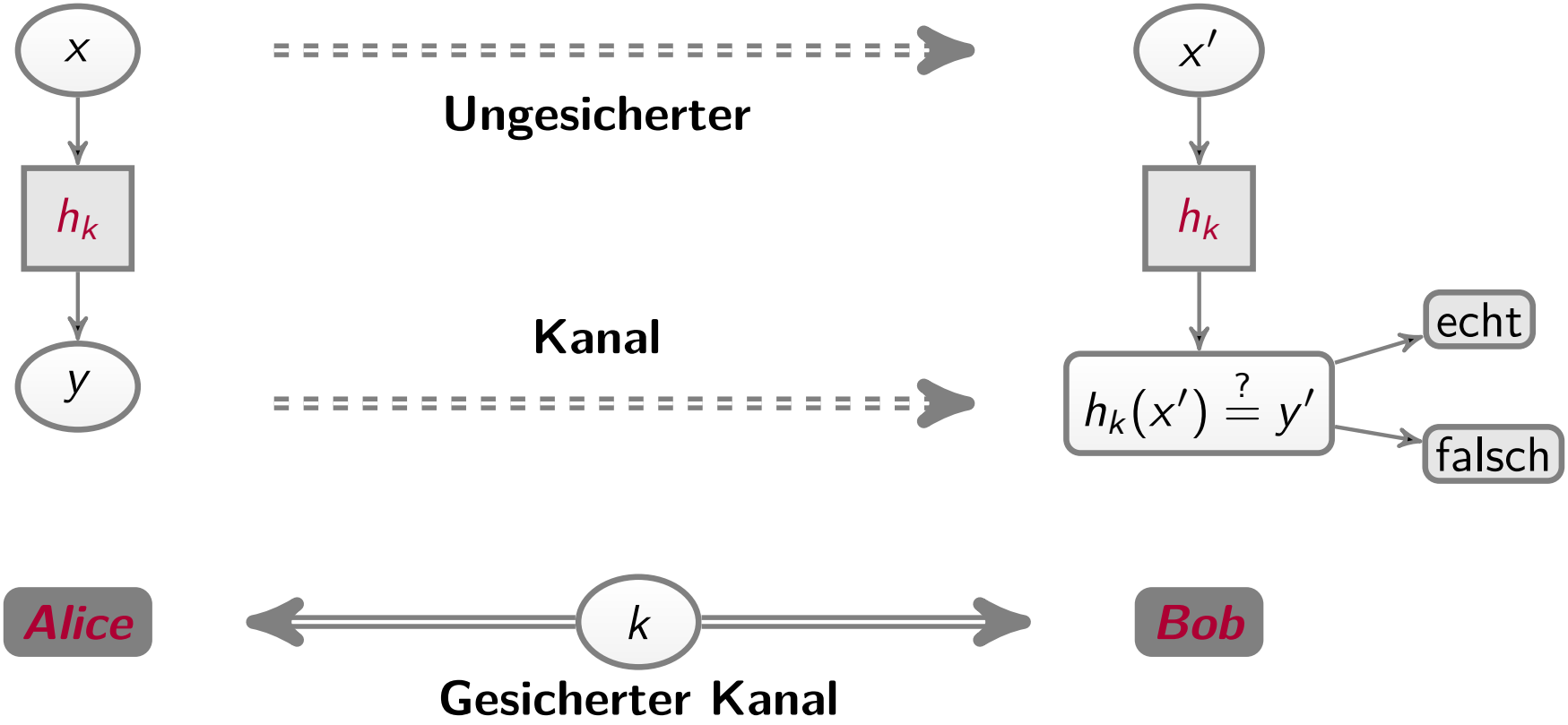
Definition

Eine ***Hashfamilie*** $\mathcal{H} = (X, Y, K, H)$ wird durch folgende Komponenten beschrieben:

- X, eine endliche oder unendliche Menge von Texten
- Y, endliche Menge aller möglichen Hashwerte, $\|Y\| \leq \|X\|$
- K, endlicher Schlüsselraum (key space), wobei jeder Schlüssel $k \in K$ eine Hashfunktion $h_k : X \to Y$ in H spezifiziert, d.h. $H = \{h_k \mid k \in K\}$

- Im folgenden werden wir die Größe $\|X\|$ des Textraumes mit n, die des Hashwertbereiches Y mit m und die des Schlüsselraumes K mit l bezeichnen
- Wir nennen dann $\mathcal{H}$ auch eine $(\mathbf{n}, \mathbf{m}, \mathbf{l})$-***Hashfamilie*** oder einen $(\mathbf{n}, \mathbf{m}, \mathbf{l})$-***MAC***

Verwendung eines MAC zur Nachrichtenauthentikation



- Hierbei ist k der symmetrische Authentikationsschlüssel und $y = h_k(x)$ der MAC-Wert für x unter k
- Indem Alice ihre Nachricht x um den Hashwert $y = h_k(x)$ ergänzt, hat Bob nicht nur die Möglichkeit, anhand von y die empfangene Nachricht x' auf Manipulationen, sondern auch ihre Herkunft zu überprüfen

Sicherheitseigenschaften von MACs

Damit ein geheimer Schlüssel k für die Authentifizierung mehrerer Nachrichten benutzt werden kann, ohne dass dies einem potentiellen Angreifer zur nichtautorisierten Berechnung von gültigen MAC-Werten verhilft, sollte folgende Bedingung erfüllt sein

Berechnungsresistenz: Auch wenn eine Reihe von unter einem Schlüssel k generierten Text-Hashwert-Paaren $(x_1, h_k(x_1)), \ldots, (x_n, h_k(x_n))$ bekannt ist, erfordert es einen immensen Aufwand, ohne Kenntnis von k ein weiteres Paar (x, y) mit $y = h_k(x)$ zu finden

Sicherheitseigenschaften von MACs

- Bei Verwendung eines berechnungsresistenten MACs ist es einem Angreifer nicht möglich, an Bob eine Nachricht x zu schicken, die Bob als von Alice stammend anerkennt
- Zu beachten ist allerdings, dass die Berechnungsresistenz nichts für den Fall aussagt, dass der Schlüssel k bekannt ist
- So kann nicht davon ausgegangen werden, dass die Funktion $h_k(x)$ bei bekanntem k die Einweg-Eigenschaft besitzt oder schwach (beziehungsweise stark) kollisionsresistent ist
- Es ist jedoch leicht zu sehen, dass es die Berechnungsresistenz erfordert, dass $h_k(x)$ bei geheimgehaltenem k zumindest schwach kollisionsresistent ist
- Dies ist etwa der Fall, wenn k im Speicher eines ausforschungssicheren Chips abgelegt wird

Verwendung eines MAC zur Versiegelung von Software

- Mithilfe eines berechnungsresistenten MACs kann der Integritätsschutz für mehrere Datensätze auf die Geheimhaltung eines Schlüssels k zurückgeführt werden
- Um die Datensätze $x_1, \ldots, x_n$ gegen unbefugt vorgenommene Veränderungen zu schützen, legt man sie zusammen mit ihren MAC-Werten $y_1 = h_k(x_1), \ldots, y_n = h_k(x_n)$ auf einem unsicheren Speichermedium ab und bewahrt den geheimen Schlüssel k an einem sicheren Ort auf
- Bei einem späteren Zugriff auf einen Datensatz x_i lässt sich dessen Unversehrtheit durch einen Vergleich von y_i mit dem Ergebnis $h_k(x_i)$ einer erneuten MAC-Berechnung überprüfen
- Da auf diese Weise ein wirksamer Schutz der Datensätze gegen Viren und andere Manipulationen erreicht wird, spricht man von einer Versiegelung der gespeicherten Datensätze

Angriffe gegen symmetrische Hashfunktionen

- Ein Angriff gegen einen MAC hat die unbefugte Berechnung von MAC-Werten zum Ziel
- Das heißt, der Angreifer versucht, MAC-Werte $h_k(x)$ ohne Kenntnis des geheimen Schlüssels k zu berechnen
- Entsprechend der Art des zur Verfügung stehenden Textmaterials lassen sich die Angriffe gegen einen MAC wie folgt klassifizieren

Impersonation:

Der Angreifer kennt nur den benutzten MAC und versucht ein Paar (x, y) mit $h_k(x) = y$ zu generieren, wobei k der (dem Angreifer unbekannte) Schlüssel ist

Substitution:

Der Angreifer versucht in Kenntnis eines Paares $(x, h_k(x))$ ein Paar (x', y') mit $x' \neq x$ und $h_k(x') = y'$ zu generieren

Angriffe gegen symmetrische Hashfunktionen

Angriff bei bekanntem Text (known-text attack):

Der Angreifer kennt für eine Reihe von Texten $x_1, \ldots, x_r$ (die er nicht selbst wählen konnte) die zugehörigen MAC-Werte $h_k(x_1), \ldots, h_k(x_r)$ und versucht, ein Paar (x', y') mit $h_k(x') = y'$ und $x' \notin \{x_1, \ldots, x_r\}$ zu generieren

Angriff bei frei wählbarem Text (chosen-text attack):

Der Angreifer kann die Texte x_i selbst wählen

Angriff bei adaptiv wählbarem Text (adaptive chosen-text attack):

Der Angreifer kann die Wahl des Textes x_i von den zuvor erhaltenen MAC-Werten $h_k(x_j)$, $j < i$, abhängig machen

Wechseln die Anwender nach jeder MAC-Wertberechnung den Schlüssel, so genügt es, dass $\mathcal{H}$ einem Impersonationsangriff widersteht

Informationstheoretische Sicherheit von MACs

Modell: Schlüssel k und Nachrichten x werden unabhängig gemäß einer Wahrscheinlichkeitsverteilung $p(k, x) = p(k)p(x)$ generiert, welche dem Angreifer bekannt ist

- Dabei nehmen wir an, dass $p(x) > 0$ und $p(k) > 0$ für alle $x \in X$ und $k \in K$ gilt
- Sei α die Wahrscheinlichkeit, mit der sich ein Angreifer bei optimaler Strategie als Alice ausgeben kann, ohne dass Bob dies bemerkt

Informationstheoretische Sicherheit von MACs

Erfolgswahrscheinlichkeit für Impersonation

- Für ein Paar (x, y) sei $p(x \mapsto y)$ die Wahrscheinlichkeit, dass ein zufällig gewählter Schlüssel den Text x auf den MAC-Wert y abbildet:

$$p(x \mapsto y) = p(y|x) = \sum_{k \in K(x,y)} p(k)$$

  wobei $K(x, y) = \{k \in K \mid h_k(x) = y\}$ alle Schlüssel enthält, die x auf y abbilden
- Bei einem Impersonationsangriff ist $p(x \mapsto y)$ also die Wahrscheinlichkeit, dass der Angreifer bei Wahl des Paares (x, y) Erfolg hat
- Deshalb bezeichnen wir diese Wahrscheinlichkeit auch mit $\alpha(x, y)$
- Schließlich ist $\alpha(x) = \max\{\alpha(x, y) \mid y \in Y\}$ die Wahrscheinlichkeit, mit der einem Angreifer bei optimaler Strategie eine Impersonation mit dem Text x gelingt
- Daher ist $\alpha = \max\{\alpha(x) \mid x \in X\}$

Informationstheoretische Sicherheit von MACs

Beispiel

- Sei $K = \{1, 2, 3\}$, $X = \{a, b, c, d\}$ und $Y = \{0, 1\}$
- Wir beschreiben H durch die zugehörige Authentikationsmatrix
- Die Zeilen und Spalten dieser Matrix werden mit den Schlüsseln $k \in K$ und den Texten $x \in X$ indiziert und ihr Eintrag in Zeile k und Spalte x ist der Wert $h_k(x)$:

		0,1	0,2	0,3	0,4
		a	b	c	d
0,25	1	0	0	0	1
0,30	2	1	1	0	1
0,45	3	0	1	1	0

- Die umrahmten Zahlen geben die Wahrscheinlichkeiten $p(x)$ bzw. $p(k)$ an

Informationstheoretische Sicherheit von MACs

Beispiel (Fortsetzung)

- Dann hat der Angreifer folgende Erfolgsaussichten $\alpha(x)$, falls er an *Bob* den Text x senden möchte

x	a	b	c	d
$p(x \mapsto 0)$	0,7	0,25	0,55	0,45
$p(x \mapsto 1)$	0,3	0,75	0,45	0,55
$\alpha(x)$	0,7	0,75	0,55	0,55

- Folglich ist $\alpha = 0{,}75$ ◁

Informationstheoretische Sicherheit von MACs

Satz

Für alle $x \in X$ ist $\alpha(x) \geq \frac{1}{m}$ und daher gilt $\alpha \geq \frac{1}{m}$

Beweis.

- Für beliebiges $x \in X$ gilt

$$\sum_{y \in Y} p(x \mapsto y) = \sum_{y \in Y} \sum_{k \in K(x,y)} p(k) = \sum_{k \in K} p(k) = 1$$

- Somit existiert für jedes $x \in X$ ein $y \in Y$ mit $p(x \mapsto y) \geq \frac{1}{m}$ und dies impliziert

$$\alpha(x) = \max_{y \in Y} p(x \mapsto y) \geq \frac{1}{m}$$

□

Informationstheoretische Sicherheit von MACs

Bemerkung

- Wie der Beweis zeigt, gilt $\alpha = \frac{1}{m}$ genau dann, wenn für alle Paare $(x, y) \in X \times Y$ folgende Gleichheit gilt:

$$\sum_{k \in K(x,y)} p(k) = \frac{1}{m}$$

- D.h. bei Gleichverteilung der Schlüssel muss in jeder Spalte der Authentikationsmatrix jeder MAC-Wert gleich oft vorkommen
- Dies lässt sich am einfachsten dadurch erreichen, dass man $K = Y$ setzt und für h_k die konstante Funktion $h_k(x) = k$ wählt

Der Entropiebegriff

Ein Maß für den Informationsgehalt

- In der Informationstheorie wird die Unsicherheit über eine Nachrichtenquelle X nach ihrer Entropie bemessen
- Dabei entspricht die Unsicherheit über X genau dem Informationsgewinn, der sich aus der Beobachtung der Quelle X ergibt
- Intuitiv ist die in einer einzelnen Nachricht x steckende Information umso größer, desto unwahrscheinlicher sie ist
- Tritt eine Nachricht x mit der Wahrscheinlichkeit $p(x) = \Pr[X = x] > 0$ auf, dann ist ihr ***Informationsgehalt*** definiert als

$$Inf_X(x) = \log_2(1/p(x)) = -\log_2 p(x)$$

- Im Fall $p(x) = 0$ sei $Inf_X(x) = 0$

Der Entropiebegriff

Ein Maß für den Informationsgehalt

- Diese Definition des Informationsgehalts ergibt sich zwangsläufig aus den beiden folgenden Axiomen:
  - Der (gemeinsame) Informationsgehalt $Inf_{X,Y}(x, y)$ von zwei Nachrichten x und y, die aus unabhängigen Quellen X und Y stammen, ist $Inf_X(x) + Inf_Y(y)$
  - Eine Nachricht x, die mit Wahrscheinlichkeit $\Pr[X = x] = 1/2$ auftritt, hat den Informationsgehalt $Inf_X(x) = 1$
- Die Einheit des Informationsgehalts ist bit (basic indissoluble information unit)
- Die Entropie von X ist nun der erwartete Informationsgehalt einer von X generierten Nachricht

Der Entropiebegriff

Definition

- Sei X eine Zufallsvariable mit Wertebereich $W(X) = \{x_1, \ldots, x_n\}$ und sei $p_i = \Pr[X = x_i]$
- Dann ist die ***Entropie*** von X definiert als

$$\mathcal{H}(X) = \sum_{i=1}^{n} p_i \, Inf_X(x_i) = \sum_{i=1}^{n} p_i \log_2(1/p_i) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

Beispiel

- Sei X eine Zufallsvariable mit der Verteilung

x_i	sonnig	leicht bewölkt	bewölkt	stark bewölkt	Regen	Schnee	Nebel
p_i	$1/4$	$1/4$	$1/8$	$1/8$	$1/8$	$1/16$	$1/16$

- Dann ergibt sich die Entropie von X zu

$$\mathcal{H}(X) = 1/4 \cdot (2+2) + 1/8 \cdot (3+3+3) + 1/16 \cdot (4+4) = 2{,}625$$

◁

Der Entropiebegriff

- Die Entropie nimmt im Fall der Gleichverteilung $p_1 = \cdots = p_n = 1/n$ den Wert $\log_2(n)$ an, während sie für jede andere Verteilung auf einer n-elementigen Menge einen Wert $\mathcal{H}(X) < \log_2(n)$ hat (siehe unten)
- Die Unsicherheit über eine Zufallsvariable X ist um so größer, je größer der Wertebereich und je gleichmäßiger die Verteilung von X ist
- Bringt X zum Beispiel nur einen einzigen Wert mit positiver Wahrscheinlichkeit hervor, dann (und nur dann) hat $\mathcal{H}(X)$ den Wert 0
- Für den Nachweis von oberen Schranken für die Entropie benutzen wir folgende Hilfsmittel aus der Analysis

Definition

- Sei $I \subseteq \mathbb{R}$ ein Intervall. Eine Funktion $f : I \to \mathbb{R}$ heißt ***konkav*** auf I, falls für alle $x \neq y \in I$ und $0 \leq t \leq 1$ gilt:

$$f(tx + (1-t)y) \quad \geq \quad tf(x) + (1-t)f(y)$$

- Gilt sogar „$>$“ anstelle von „$\geq$“, so heißt f ***streng konkav*** auf I
- Im Falle von „$<$“ bzw. „$\leq$“ heißt f ***(streng) konvex*** auf I

Der Entropiebegriff

Beispiel

Die Funktion $f(x) = \log_2(x)$ ist streng konkav auf $(0, \infty)$ ◁

Für den Beweis des nächsten Satzes benötigen wir die Jensensche Ungleichung, die wir ohne Beweis angeben

Jensensche Ungleichung

- Sei f eine streng konkave Funktion auf I und seien $0 < a_1, \ldots, a_n < 1$ reelle Zahlen mit $\sum_{i=1}^{n} a_i = 1$
- Dann gilt für alle $x_1, \ldots, x_n \in I$,

$$f\left(\sum_{i=1}^{n} a_i x_i\right) \quad \geq \quad \sum_{i=1}^{n} a_i f(x_i)$$

- Im Falle einer streng konvexen Funktion f gilt $\leq$ anstelle von $\geq$
- Dabei gilt Gleichheit genau dann, wenn alle x_i den gleichen Wert haben

Der Entropiebegriff

Satz

- Sei X eine Zufallsvariable mit Wertebereich $W(X) = \{x_1, \ldots, x_n\}$ und Verteilung $p_i = \Pr[X = x_i]$ für $i = 1, \ldots, n$
- Dann gilt $H(X) \leq \log_2(n)$, wobei Gleichheit genau im Fall $p_i = 1/n$ für $i = 1, \ldots, n$ eintritt

Beweis.

- Aufgrund der Jensenschen Ungleichung gilt

$$\mathcal{H}(X) = \sum_{i=1}^{n} p_i \log_2(1/p_i) \leq \log_2 \sum_{i=1}^{n} (p_i/p_i) = \log_2 n,$$

  wobei Gleichheit genau im Fall $1/p_1 = \cdots = 1/p_n$ eintritt
- Letzteres ist mit der Bedingung $p_i = 1/n$ für $i = 1, \ldots, n$ gleichbedeutend

□

Informationstheoretische Sicherheit von MACs

Das folgende Lemma benötigen wir für den Beweis des nächsten Satzes

Lemma

- Sei $\mathcal{X}$ eine Zufallsvariable mit endlichem Wertebereich $W(\mathcal{X}) \subseteq \mathbb{R}^+$
- Dann gilt $\log E(\mathcal{X}) \geq E(\log \mathcal{X})$

Beweis.

- Sei $W(\mathcal{X}) = \{x_1, \ldots, x_n\}$ und für $i = 1, \ldots, n$ sei $p_i = \Pr[\mathcal{X} = x_i]$
- Da die Funktion $x \mapsto \log_2 x$ konkav ist, folgt mit der Jensenschen Ungleichung

$$\log E(\mathcal{X}) = \log_2(\sum p_i x_i) \geq \sum p_i \log_2 x_i = E(\log \mathcal{X})$$

□

Die bedingte Entropie

Zudem benötigen wir noch den Begriff der bedingten Entropie

Definition. Seien X, Y Zufallsvariablen

Die ***bedingte Entropie*** von X unter Y ist definiert als

$$\mathcal{H}(X|Y) = \sum_{y \in W(Y)} p(y)\mathcal{H}(X|y) = \sum_{y} p(y) \sum_{x} p(x|y) \log_2(1/p(x|y)),$$

wobei $X|y$ die Zufallsvariable mit der Verteilung $p_y(x) = p(x|y)$ ist

Informationstheoretische Sicherheit von MACs

Satz

- Für jeden MAC (X, Y, K, H) gilt:

$$\alpha \geq \frac{1}{2^{H(\mathcal{K})-H(\mathcal{K}|\mathcal{X},\mathcal{Y})}} \geq 1/I$$

- Hierbei sind $\mathcal{X}, \mathcal{Y}, \mathcal{K}$ Zufallsvariablen, die die Verteilungen der Nachrichten, der MAC-Werte und der Schlüssel beschreiben

Der Wert von α kann also um so kleiner werden, je gleichmäßiger die Schlüsselverteilung ist und je mehr Information die Beobachtung eines gültigen Paares (x, y) über den Schlüssel liefert

Informationstheoretische Sicherheit von MACs

Satz

Für jeden MAC (X, Y, K, H) gilt $\alpha \geq 1/2^{H(\mathcal{K})-H(\mathcal{K}|\mathcal{X},\mathcal{Y})} \geq 1/I$

Beweis.

- Wegen $\alpha = \max_{x,y} \alpha(x,y)$ ist $E(\alpha(\mathcal{X},\mathcal{Y})) = \sum_{x,y} p(x,y)\alpha(x,y) \leq \alpha$
- Dabei ist $E(\alpha(\mathcal{X},\mathcal{Y}))$ die Erfolgswahrscheinlichkeit eines (probabilistischen) Angreifers, der das Paar (x,y) gemäß der Verteilung $(\mathcal{X},\mathcal{Y})$ wählt
- Somit folgt unter Anwendung von obigem Lemma

$$\begin{aligned} \log \alpha &\geq \log E(\alpha(\mathcal{X},\mathcal{Y})) \geq E(\log \alpha(\mathcal{X},\mathcal{Y})) \\ &= \sum_{x,y} \underbrace{p(x,y)}_{p(x)p(y|x)} \underbrace{\log \alpha(x,y)}_{\log p(y|x) = -\log \frac{1}{p(y|x)}} = -H(\mathcal{Y}|\mathcal{X}) \end{aligned}$$

Informationstheoretische Sicherheit von MACs

Beweis.

- Somit folgt unter Anwendung von obigem Lemma

$$\log \alpha \geq -H(\mathcal{Y} \mid \mathcal{X})$$

- Zudem gilt

$$H(\mathcal{K}, \mathcal{Y}, \mathcal{X}) = H(\mathcal{X}) + H(\mathcal{Y} \mid \mathcal{X}) + H(\mathcal{K} \mid \mathcal{X}, \mathcal{Y})$$

und

$$H(\mathcal{K}, \mathcal{Y}, \mathcal{X}) = \underbrace{H(\mathcal{K}, \mathcal{X})}_{= H(\mathcal{K}) + H(\mathcal{X})} + \underbrace{H(\mathcal{Y} \mid \mathcal{K}, \mathcal{X})}_{= 0}$$

- Daher folgt $H(\mathcal{Y} \mid \mathcal{X}) = H(\mathcal{K}) - H(\mathcal{K} \mid \mathcal{X}, \mathcal{Y})$ und somit $\log \alpha \geq H(\mathcal{K} \mid \mathcal{X}, \mathcal{Y}) - H(\mathcal{K})$
- Dies ist äquivalent zu $\alpha \geq 1/2^{H(\mathcal{K}) - H(\mathcal{K} \mid \mathcal{X}, \mathcal{Y})}$

□

Informationstheoretische Sicherheit von MACs

Beispiel

- Sei $K = \{1, 2, 3\}$, $X = \{a, b, c, d\}$ und $Y = \{0, 1\}$
- Wir beschreiben H durch die zugehörige Authentikationsmatrix
- Die Zeilen und Spalten dieser Matrix werden mit den Schlüsseln $k \in K$ und den Texten $x \in X$ indiziert und ihr Eintrag in Zeile k und Spalte x ist der Wert $h_k(x)$:

		0,1	0,2	0,3	0,4
		a	b	c	d
0,25	1	0	0	0	1
0,30	2	1	1	0	1
0,45	3	0	1	1	0

- Die umrahmten Zahlen geben die Wahrscheinlichkeiten $p(x)$ bzw. $p(k)$ an

Beispiel (Fortsetzung)

- Es gilt

$$H(\mathcal{K}) = \sum_k p(k) \log \frac{1}{p(k)} = 0{,}45 \cdot 1{,}152 + 0{,}3 \cdot 1{,}737 + 0{,}25 \cdot 2{,}0 = 1{,}54$$

- Um $H(\mathcal{K}|\mathcal{X},\mathcal{Y})$ zu bestimmen, benötigen wir die gemeinsame Verteilung von $\mathcal{X},\mathcal{Y}$ sowie die bedingten Verteilungen $\mathcal{K}_{x,y}$ für alle Paare $(x,y) \in X \times Y$:

(x,y)	$(a,0)$	$(a,1)$	$(b,0)$	$(b,1)$	$(c,0)$	$(c,1)$	$(d,0)$	$(d,1)$
$p(x,y)$	0,07	0,03	0,05	0,15	0,165	0,135	0,18	0,22
$p(1\|x,y)$	$\frac{5}{14}$	0	1	0	$\frac{5}{11}$	0	0	$\frac{5}{11}$
$p(2\|x,y)$	0	1	0	$\frac{2}{5}$	$\frac{6}{11}$	0	0	$\frac{6}{11}$
$p(3\|x,y)$	$\frac{9}{14}$	0	0	$\frac{3}{5}$	0	1	1	0
$H(\mathcal{K}\|x,y)$	$\approx 0{,}94$	0	0	$\approx 0{,}97$	$\approx 0{,}99$	0	0	$\approx 0{,}99$

- Hierbei gilt $p(x,y) = p(x)p(y|x) = p(x)p(x \mapsto y)$

Beispiel (Schluss)

- Somit ist

$$H(\mathcal{K}|\mathcal{X},\mathcal{Y}) = \sum_{x,y} p(x,y) H(\mathcal{K}|x,y) \approx 0{,}52$$

und wir erhalten die untere Schranke

$$\alpha \geq \frac{1}{2^{H(\mathcal{K})-H(\mathcal{K}|\mathcal{X},\mathcal{Y})}} \approx \frac{1}{2^{1{,}54-0{,}52}} = \frac{1}{2^{1{,}02}} \approx 0{,}493$$

◁

Erfolgswahrscheinlichkeit für Substitution

- Bezeichne β die Wahrscheinlichkeit, mit der ein MAC-Angreifer bei optimaler Strategie eine von Alice gesendete Nachricht x durch eine andere Nachricht x' ersetzen kann, ohne dass Bob dies bemerkt
- Dabei gehen wir davon aus, dass der Angreifer keinen Einfluss auf die Wahl der von Alice gesendeten Nachricht x hat
- Falls der Angreifer ein von Alice gesendetes Paar (x, y) durch das Paar (x', y') ersetzt, ist seine Erfolgswahrscheinlichkeit gleich der bedingten Wahrscheinlichkeit

$$p(x' \mapsto y' \,|\, x \mapsto y) = \frac{p(x \mapsto y, x' \mapsto y')}{p(x \mapsto y)} = \frac{\sum_{k \in K(x,y,x',y')} p(k)}{\sum_{k \in K(x,y)} p(k)},$$

  dass ein zufällig gewählter Schlüssel k den Text x' auf y' abbildet, wenn bereits bekannt ist, dass $h_k(x) = y$ ist
- Hierbei ist $K(x, y, x', y') = \{k \in K \mid h_k(x) = y \wedge h_k(x') = y'\}$

Erfolgswahrscheinlichkeit für Substitution

- Falls Alice also das Paar (x, y) sendet, so ist die maximale Erfolgswahrscheinlichkeit des Angreifers

$$\beta(x, y) := \max_{x' \neq x, y'} p(x' \mapsto y' \,|\, x \mapsto y)$$

- Man beachte, dass $\beta(x, y)$ nur im Fall $p(x, y) > 0$ definiert ist
- Da der Angreifer keinen Einfluss auf die Wahl von (x, y) hat, ist β gleich dem Erwartungswert von $\beta(x, y)$ unter der Verteilung $p(x, y)$, mit der Alice diese Paare generiert
- Somit erhalten wir

$$\beta = E(\beta(\mathcal{X}, \mathcal{Y})) = \sum_{x \in X, y \in Y} p(x, y)\beta(x, y)$$

- Wegen $p(x, y) = p(x)p(x \mapsto y)$ können wir β unter Verwendung von

$$\beta'(x, y) = \beta(x, y)p(x \mapsto y) = \max_{x' \neq x, y'} p(x' \mapsto y', x \mapsto y)$$

auch mittels der Formel $\beta = \sum_{x \in X} p(x) \sum_{y \in Y} \beta'(x, y)$ berechnen

Erfolgswahrscheinlichkeit für Substitution

Beispiel

- Sei $K = \{1, 2, 3\}$, $X = \{a, b, c, d\}$ und $Y = \{0, 1\}$
- Wir beschreiben H durch die zugehörige Authentikationsmatrix
- Die Zeilen und Spalten dieser Matrix werden mit den Schlüsseln $k \in K$ und den Texten $x \in X$ indiziert und ihr Eintrag in Zeile k und Spalte x ist der Wert $h_k(x)$:

		0,1	0,2	0,3	0,4
		a	b	c	d
0,25	1	0	0	0	1
0,30	2	1	1	0	1
0,45	3	0	1	1	0

- Die umrahmten Zahlen geben die Wahrscheinlichkeiten $p(x)$ bzw. $p(k)$ an

Erfolgswahrscheinlichkeit für Substitution

Beispiel (Fortsetzung)

(x,y)	$p(x' \mapsto y', x \mapsto y)$								$\beta'(x,y)$	$p(x \mapsto y)$	$\beta(x,y)$
	$(a,0)$	$(a,1)$	$(b,0)$	$(b,1)$	$(c,0)$	$(c,1)$	$(d,0)$	$(d,1)$			
$(a,0)$			0,25	**0,45**	0,25	**0,45**	**0,45**	0,25	0,45	0,7	0,643
$(a,1)$			0	**0,3**	**0,3**	0	0	**0,3**	0,3	0,3	1
$(b,0)$	**0,25**	0			**0,25**	0	0	**0,25**	0,25	0,25	1
$(b,1)$	**0,45**	0,3			0,3	**0,45**	**0,45**	0,3	0,45	0,75	0,6
$(c,0)$	0,25	0,3	0,25	0,3			0	**0,55**	0,55	0,55	1
$(c,1)$	**0,45**	0	0	**0,45**			**0,45**	0	0,45	0,45	1
$(d,0)$	**0,45**	0	0	**0,45**	0	**0,45**			0,45	0,45	1
$(d,1)$	0,25	0,3	0,25	0,3	**0,55**	0			0,55	0,55	1

- Die optimalen Wahlmöglichkeiten des Angreifers, ein Paar (x, y) durch ein anderes Paar (x', y') zu ersetzen, sind in der Tabelle fett gedruckt

Erfolgswahrscheinlichkeit für Substitution

Beispiel (Schluss)

- Für β erhalten wir somit den Wert

$$\begin{aligned}\beta &= \sum_{x\in X} p(x) \sum_{y\in Y} \beta'(x,y) \\ &= 0{,}1(0{,}45+0{,}3)+0{,}2(0{,}25+0{,}45)+0{,}3(0{,}55+0{,}45) \\ &\quad + 0{,}4(0{,}45+0{,}55) \\ &= 0{,}915\end{aligned}$$

◁

Erfolgswahrscheinlichkeit für Substitution

Als nächstes zeigen wir für β die gleiche untere Schranke wie für α

Satz

Für alle $(x, y) \in X \times Y$ mit $p(x, y) > 0$ ist $\beta(x, y) \geq \frac{1}{m}$ und somit $\beta \geq \frac{1}{m}$

Beweis.

- Sei $(x, y) \in X \times Y$ ein Paar mit $p(x, y) > 0$
- Dann gilt für beliebige $x' \in X - \{x\}$

$$\sum_{y' \in Y} p(x' \mapsto y' \,|\, x \mapsto y) = \frac{\sum_{y' \in Y} \sum_{k \in K(x', y'; x, y)} p(k)}{\sum_{k \in K(x,y)} p(k)} = 1$$

- Somit existiert ein $y' \in Y$ mit $p(x' \mapsto y' \,|\, x \mapsto y) \geq \frac{1}{m}$ und dies impliziert

$$\beta(x, y) \quad = \quad \max_{x' \neq x, y'} p(x' \mapsto y' \,|\, x \mapsto y) \quad \geq \quad \frac{1}{m}$$

- Also gilt $\beta = \sum_{x \in X, y \in Y} p(x, y)\beta(x, y) \geq \frac{1}{m} \sum_{x \in X, y \in Y} p(x, y) = \frac{1}{m}$ □

Erfolgswahrscheinlichkeit für Substitution

Beispiel

- Sei $X = Y = \{0, 1, 2\} = \mathbb{Z}_3$ und sei $K = \mathbb{Z}_3 \times \mathbb{Z}_3$
- Für $k = (a, b) \in K$ und $x \in X$ sei

$$h_k(x) = ax + b \bmod 3$$

- Die zugehörige Authentikationsmatrix ist

	0	1	2
$(0,0)$	0	0	0
$(0,1)$	1	1	1
$(0,2)$	2	2	2
$(1,0)$	0	1	2
$(1,1)$	1	2	0
$(1,2)$	2	0	1
$(2,0)$	0	2	1
$(2,1)$	1	0	2
$(2,2)$	2	1	0

- Wir nehmen an, dass der Schlüssel unter Gleichverteilung gewählt wird

Erfolgswahrscheinlichkeit für Substitution

Beispiel (Fortsetzung)

- Ersetzt der Angreifer ein Paar (x, y) durch ein Paar (x', y') mit $x' \neq x$, so wird dieses Paar von genau einem der 3 infrage kommenden Schlüssel akzeptiert
- Dies liegt daran, dass in je 2 Spalten der Authentikationsmatrix jedes MAC-Wertepaar genau einmal vorkommt
- Folglich ist $p(x' \mapsto y' \,|\, x \mapsto y) = 1/3$ und somit hat β den optimalen Wert $\beta = 1/3$ $\triangleleft$

Erfolgswahrscheinlichkeit für Substitution

Lemma

In einem MAC (X, Y, K, H) mit $\beta = \frac{1}{m}$ gilt für alle Doppelpaare (x, y, x', y') mit $x \neq x'$ die Gleichheit

$$p(x' \mapsto y' \,|\, x \mapsto y) = 1/m$$

Beweis.

- Wir setzen zunächst voraus, dass $p(x \mapsto y) > 0$ für alle Paare $(x, y) \in X \times Y$ gilt
- Würde nun für ein Doppelpaar (x, y, x', y') mit $x \neq x'$

  $$p(x' \mapsto y' \,|\, x \mapsto y) > 1/m$$

  gelten, dann wäre auch

  $$\beta(x, y) = \max_{x' \neq x, y'} p(x' \mapsto y' \,|\, x \mapsto y) > 1/m$$

Erfolgswahrscheinlichkeit für Substitution

Beweis (Fortsetzung).

- Da für alle Paare (u, v) mit $p(u \mapsto v) > 0$ nach obigem Satz die Ungleichung $\beta(u, v) \geq 1/m$ gilt und $p(x, y) = p(x)p(x \mapsto y) > 0$ ist, würde hieraus

$$\beta = \sum_{u \in X, v \in Y} p(u, v)\beta(u, v) = p(x, y)\underbrace{\beta(x, y)}_{>1/m} + \sum_{(u,v) \neq (x,y)} p(u, v)\underbrace{\beta(u, v)}_{\geq 1/m} > 1/m$$

  folgen, was im Widerspruch zur Voraussetzung des Satzes steht

- Ist andererseits

$$p(x' \mapsto y' \,|\, x \mapsto y) < 1/m,$$

  muss wegen

$$\sum_{y'' \in Y} p(x' \mapsto y'' \,|\, x \mapsto y) = 1$$

  auch ein MAC-Wert y'' mit $p(x' \mapsto y'' \,|\, x \mapsto y) > 1/m$ existieren, was wir bereits widerlegt haben

Erfolgswahrscheinlichkeit für Substitution

Beweis (Schluss).

- Es bleibt zu zeigen, dass $p(x \mapsto y) > 0$ für alle Paare $(x, y) \in X \times Y$ gilt
- Wäre $p(x \mapsto y) = 0$, so würde für ein beliebiges Paar (u, v) mit $p(u \mapsto v) > 0$ auch $p(x \mapsto y \,|\, u \mapsto v) = 0 < 1/m$ sein, was wir bereits widerlegt haben $\square$

Erfolgswahrscheinlichkeit für Substitution

Satz

Ein MAC (X, Y, K, H) erfüllt $\beta = \frac{1}{m}$ genau dann, wenn

$$p(x \mapsto y, x' \mapsto y') = 1/m^2$$

für alle Doppelpaare (x, y, x', y') mit $x \neq x'$ gilt

Beweis.

- Sei (X, Y, K, H) ein MAC mit $\beta = \frac{1}{m}$
- Nach obigem Lemma impliziert dies, dass für alle Doppelpaare (x, y, x', y') mit $x \neq x'$ gilt,

$$p(x' \mapsto y' \,|\, x \mapsto y) = 1/m$$

- Dies impliziert nun

$$p(x' \mapsto y') = \sum_y p(x \mapsto y) p(x' \mapsto y' \,|\, x \mapsto y) = 1/m$$

und daher

$$p(x \mapsto y, x' \mapsto y') = p(x' \mapsto y') p(x \mapsto y \,|\, x' \mapsto y') = 1/m^2$$

Erfolgswahrscheinlichkeit für Substitution

Satz

Ein MAC (X, Y, K, H) erfüllt $\beta = \frac{1}{m}$ genau dann, wenn

$$p(x \mapsto y, x' \mapsto y') = 1/m^2$$

für alle Doppelpaare (x, y, x', y') mit $x \neq x'$ gilt

Beweis (Schluss).

- Umgekehrt rechnet man leicht nach, dass die Bedingung $\beta = \frac{1}{m}$ erfüllt ist, wenn für alle Doppelpaare (x, y, x', y') mit $x \neq x'$ die Gleichheit $p(x \mapsto y, x' \mapsto y') = 1/m^2$ gilt □

Erfolgswahrscheinlichkeit für Substitution

Bemerkung

- Nach obigem Satz gilt $\beta = \frac{1}{m}$ genau dann, wenn für alle Doppelpaare (x, y, x', y') mit $x \neq x'$ gilt,

$$p(x \mapsto y, x' \mapsto y') = \sum_{k \in K(x,y,x',y')} p(k) = \frac{1}{m^2}$$

- D.h. bei Gleichverteilung der Schlüssel gilt $\beta = \frac{1}{m}$ genau dann, wenn in je zwei Spalten der Authentikationsmatrix jedes MAC-Wertepaar gleich oft vorkommt

Konstruktion von 2-universalen MACs

Ab jetzt setzen wir voraus, dass der Schlüssel unter Gleichverteilung gewählt wird, d.h. es gilt $p(k) = \frac{1}{\|K\|}$ für alle $k \in K$

Definition

Ein MAC (X, Y, K, H) heißt 2-***universal***, falls für alle $x, x' \in X$ mit $x \neq x'$ und alle $y, y' \in Y$ gilt:

$$\|K(x, y, x', y')\| = \frac{\|K\|}{m^2}$$

Ein MAC (X, Y, K, H) ist also genau dann 2-universal, wenn für alle Textpaare $x, x' \in X$ mit $x \neq x'$ jedes MAC-Wertpaar $y, y' \in Y$ mit Wk $1/m^2$ auftritt

Konstruktion von 2-universalen MACs

Bemerkung

- Bei der Konstruktion von 2-universalen MACs spielt der Parameter $\lambda = \frac{\|K\|}{m^2}$ eine wichtige Rolle
- Da λ notwendigerweise positiv und ganzzahlig ist, muss insbesondere $\|K\| \geq m^2$ gelten
- Im Folgenden nennen wir einen 2-universalen (n, m, l)-MAC mit $\lambda = l/m^2$ kurz einen (n, m, l, λ)-MAC

Konstruktion von 2-universalen MACs

Beispiel

- Wir betrachten den MAC (X, Y, K, H) mit $X = \{0, 1, 2, 3\}$, $Y = \{0, 1, 2\}$, $K = \{0, 1, \ldots, 8\}$, wobei H durch folgende Authentikationsmatrix beschrieben wird:

	0	1	2	3
0	0	0	0	0
1	1	1	1	0
2	2	2	2	0
3	0	1	2	1
4	1	2	0	1
5	2	0	1	1
6	0	2	1	2
7	1	0	2	2
8	2	1	0	2

- Da in je zwei Spalten jedes MAC-Wertepaar genau einmal vorkommt, ist (X, Y, K, H) ein $(4, 3, 9, 1)$-MAC ◁

Konstruktion von 2-universalen MACs

- Auf Grund obiger Bemerkung ist klar, dass ein MAC bei gleichverteilten Schlüsseln genau dann die Bedingung $\beta = \frac{1}{m}$ erfüllt, wenn er 2-universal ist
- In diesem Fall nimmt auch α den optimalen Wert $\frac{1}{m}$ an
- Der nächste Satz zeigt eine einfache Konstruktionsmöglichkeit von 2-universalen MACs mit dem Parameterwert $\lambda = 1$

Satz

- Sei p prim und für $a, b, x \in \mathbb{Z}_p$ sei

$$h_{a,b}(x) = ax + b \bmod p$$

- Dann ist (X, Y, K, H) mit $X = Y = \mathbb{Z}_p$ und $K = \mathbb{Z}_p \times \mathbb{Z}_p$ ein $(p, p, p^2, 1)$-MAC

Konstruktion von 2-universalen MACs

Beweis.

- Wir müssen zeigen, dass $K(x, y, x', y')$ für jedes Doppelpaar (x, y, x', y') mit $x \neq x'$ genau einen Schlüssel enthält
- Ein Schlüssel (a, b) gehört genau dann zu dieser Menge, wenn er die beiden Kongruenzen

$$\begin{aligned} ax + b &\equiv_p y, \\ ax' + b &\equiv_p y' \end{aligned}$$

  erfüllt
- Da dies jedoch nur auf den Schlüssel (a, b) mit

$$\begin{aligned} a &= (y - y')(x - x')^{-1} \bmod p, \\ b &= y - x(y - y')(x - x')^{-1} \bmod p \end{aligned}$$

  zutrifft, folgt $\|K(x, y, x', y')\| = 1$

□

Konstruktion von 2-universalen MACs

- Die Hashfunktionen des vorigen Satzes erfüllen wegen $n = m = p$ nicht die Kompressionseigenschaft
- Zwar lässt sich n noch geringfügig von p auf $p + 1$ (und somit der Quotient m/n von 1 auf $\frac{p}{p+1}$) verkleinern, ohne K und Y zu verändern (siehe Übungen)
- Wie der nächste Satz zeigt, lässt sich eine stärkere Kompression mit dem Parameterwert $\lambda = 1$ jedoch nicht realisieren

Satz

Für einen $(n, m, l, 1)$-MAC gilt

$$n \leq m + 1$$

und somit $l = m^2 \geq (n-1)^2$ sowie $m/n \geq \frac{m}{m+1} (\approx 1)$

Konstruktion von 2-universalen MACs

Satz

Für einen $(n, m, l, 1)$-MAC gilt

$$n \leq m + 1$$

und somit $l = m^2 \geq (n-1)^2$ sowie $m/n \geq \frac{m}{m+1} (\approx 1)$

Beweis.

- O.B.d.A. sei $K = \{1, \ldots, l\}$ und $Y = \{1, \ldots, m\}$
- Es ist leicht zu sehen, dass eine (bijektive) Umbenennung $\pi \colon Y \to Y$ der MAC-Werte in einer einzelnen Spalte der Authentikationsmatrix A wieder auf einen 2-universalen MAC führt
- Also können wir annehmen, dass die erste Zeile der Authentikationsmatrix A nur Einsen enthält

Konstruktion von 2-universalen MACs

Beweis (Schluss)

- Da A 2-universal ist, gilt:
  - In jeder Zeile $i = 2, \ldots, m^2$ kommt höchstens eine Eins vor
  - Jede Spalte j enthält eine Eins in Zeile 1 und $m-1$ Einsen in den übrigen Zeilen
- Da in den Zeilen $i = 2, \ldots, m^2$ insgesamt genau $n(m-1)$ Einsen vorkommen, folgt

$$\underbrace{\text{Anzahl der Zeilen}}_{m^2} \quad \geq \quad \underbrace{\text{Anzahl der Zeilen mit einer Eins}}_{1+n(m-1)},$$

was $m^2 - 1 \geq n(m-1)$ bzw. $n \leq m+1$ impliziert $\square$

Konstruktion von 2-universalen MACs

- Der nächste Satz liefert 2-universale MACs mit beliebig kleinem Kompressionsquotienten m/n
- Für den Beweis benötigen wir das folgende Lemma

Lemma

- Sei A eine $(k \times \ell)$-Matrix über einem endlichen Körper $\mathbb{F}$, deren k Zeilen linear unabhängig sind
- Dann besitzt das lineare Gleichungssystem

$$Ax = y$$

für jedes $y \in \mathbb{F}^k$ genau $\|\mathbb{F}\|^{\ell-k}$ Lösungen $x \in \mathbb{F}^\ell$

Beweis.

Siehe Übungen □

Konstruktion von 2-universalen MACs

Satz

- Sei p prim und für $x = (x_1, \ldots, x_d) \in \{0,1\}^d$ und $k = (k_1, \ldots, k_d) \in \mathbb{Z}_p^d$ sei

$$h_k(x) = kx = \sum_{i=1}^{d} k_i x_i \bmod p$$

- Dann ist (X, Y, K, H) mit $X = \{0,1\}^d - \{0^d\}$, $Y = \mathbb{Z}_p$ und $K = \mathbb{Z}_p^d$ ein $(2^d - 1, p, p^d, p^{d-2})$-MAC

Beweis.

- Wir müssen zeigen, dass die Größe von $K(x, y, x', y')$ für alle Doppelpaare (x, y, x', y') mit $x \neq x'$ konstant ist

Konstruktion von 2-universalen MACs

Beweis (Fortsetzung)

- Es gilt

$$\begin{aligned} k \in K(x, y, x', y') &\Leftrightarrow h_k(x) = y \ \wedge \ h_k(x') = y' \\ &\Leftrightarrow k \cdot x = y \ \wedge \ k \cdot x' = y' \end{aligned}$$

- Fassen wir $x = x_1 \cdots x_d$ und $x' = x'_1 \cdots x'_d$ zu einer Matrix A zusammen, so ist dies äquivalent zu

$$\begin{pmatrix} x_1 \cdots x_d \\ x'_1 \cdots x'_d \end{pmatrix} \begin{pmatrix} k_1 \\ \vdots \\ k_d \end{pmatrix} = \begin{pmatrix} y \\ y' \end{pmatrix}$$

- Da die beiden Zeilen von A verschieden und damit linear unabhängig sind, folgt mit obigem Lemma, dass genau $\|K(x, y, x', y')\| = p^{d-2}$ Schlüssel $k = (k_1, \ldots, k_d)$ mit dieser Eigenschaft existieren $\square$

Konstruktion von 2-universalen MACs

Bemerkung

- Obige Konstruktion liefert einen λ-Wert von $\frac{\|K\|}{m^2} = p^{d-2}$
- Durch Erweiterung von X auf eine geeignete Teilmenge $X' \subseteq \mathbb{Z}_p^d$ lässt sich der Textraum von $2^d - 1$ auf $\frac{p^d-1}{p-1}$ vergrößern (siehe Übungen)
- Dies führt auf einen beliebig kleinen Kompressionsquotienten

  $$\frac{m}{n} = \frac{p(p-1)}{p^d - 1} \approx p^{2-d}$$

  bei einem λ-Wert von $\lambda = p^{d-2}$
- Wie der nächste Satz zeigt, lässt sich dies nicht mit einem kleineren λ-Wert (bzw. nicht mit einer kleineren Schlüssellänge) erreichen

Konstruktion von 2-universalen MACs

Für den Beweis des nächsten Satzes benötigen wir folgendes Lemma

Lemma

Für beliebige reelle Zahlen $b_1, \ldots, b_m \in \mathbb{R}$ gilt $\left(\sum_{i=1}^{m} b_i\right)^2 \leq m \sum_{i=1}^{m} b_i^2$

Beweis.

- Da die Funktion $x \mapsto x^2$ konvex ist, folgt mit der Jensenschen Ungleichung $(\sum b_i/m)^2 \leq \sum b_i^2/m$
- Folglich ist $\left(\sum b_i\right)^2 = m^2 \underbrace{\left(\sum b_i/m\right)^2}_{\leq \sum b_i^2/m} \leq m \sum b_i^2$

□

Jensensche Ungleichung

Für eine konkave Funktion f und $a_1, \ldots, a_n \in (0,1)$ mit $\sum_{i=1}^{n} a_i = 1$ ist

$$f\left(\sum_{i=1}^{n} a_i x_i\right) \quad \geq \quad \sum_{i=1}^{n} a_i f(x_i)$$

Konstruktion von 2-universalen MACs

Satz

Für jeden (n, m, l, λ)-MAC gilt

$$\underbrace{\lambda m^2}_{=l} \geq n(m-1)+1 \quad \text{und somit} \quad m/n \geq \underbrace{(m-1)/m(\lambda - 1/m^2)}_{\approx 1/\lambda}$$

Beweis.

- O.B.d.A. können wir wieder $K = \{k_1, \ldots, k_l\}$ und $Y = \{1, \ldots, m\}$ annehmen, und dass die erste Zeile der Authentikationsmatrix nur aus Einsen besteht
- Für jede Zeile $i = 1, \ldots, l$ bezeichne e_i die Anzahl der Einsen in dieser Zeile (also $e_1 = n$)
- Da in jeder Spalte jeder MAC-Wert genau λm-mal vorkommt, gilt

$$\sum_{i=1}^{l} e_i = \lambda nm \quad \text{und} \quad \sum_{i=2}^{l} e_i = \lambda nm - n = n(\lambda m - 1)$$

Konstruktion von 2-universalen MACs

Beweis (Fortsetzung).

- Sei $z = \sum_{i=2}^{l} z_i$, wobei z_i die Anzahl von Spaltenpaaren (j, j') mit $j \neq j'$ und $h_{k_i}(x_j) = h_{k_i}(x_{j'}) = 1$ ist
- Dann folgt

$$z = \sum_{i=2}^{l} z_i = \sum_{i=2}^{l} e_i(e_i - 1) = \sum_{i=2}^{l} e_i^2 - \sum_{i=2}^{l} e_i = \sum_{i=2}^{l} e_i^2 - n(\lambda m - 1)$$

- Mit obigem Lemma ergibt sich

$$\sum_{i=2}^{l} e_i^2 \geq \frac{\left(\sum_{i=2}^{l} e_i\right)^2}{l-1} = \frac{(n(\lambda m - 1))^2}{l-1}$$

- Da andererseits in jedem Spaltenpaar das MAC-Wertepaar $(1,1)$ in genau λ Zeilen vorkommt (genauer: einmal in Zeile 1 und $(\lambda - 1)$-mal in den Zeilen $i = 2, \ldots, l$), und da $n(n-1)$ solche Spaltenpaare existieren, ergibt sich andererseits die Gleichung

$$z = (\lambda - 1)n(n-1)$$

Konstruktion von 2-universalen MACs

Beweis (Schluss).

- Somit erhalten wir

$$(\lambda-1)n(n-1) = z = \sum_{i=2}^{l} e_i^2 - n(\lambda m-1) \geq \frac{(n(\lambda m-1))^2}{l-1} - n(\lambda m-1)$$

und daher folgt

$$\begin{aligned}
&((\lambda-1)n(n-1) + n(\lambda m-1))(\lambda m^2-1) \geq (n(\lambda m-1))^2 \\
\Rightarrow\quad &(\lambda n - n - \lambda + \lambda m)(\lambda m^2-1) \geq n(\lambda m-1)^2 \\
\Rightarrow\quad &-\lambda^2 m^2 + \lambda^2 m^3 \geq \lambda n m^2 + \lambda n - \lambda + \lambda m - 2\lambda n m \\
\Rightarrow\quad &\lambda^2(m^3-m^2) \geq \lambda(n(m-1)^2 + m - 1) \\
\Rightarrow\quad &\lambda m^2 \geq n(m-1) + 1
\end{aligned}$$

□

CBC-MACs

- Als Basis für die Konstruktion eines MAC kann auch ein symmetrisches Kryptosystem dienen
- Sei (M, C, K, E, D) ein symmetrisches Kryptosystem mit $M = C = \{0,1\}^t$
- Zudem sei $IV := 0^t$ und sei $k \in K$ ein geheimer Schlüssel
- Sei y eine Funktion für den Preprocessing-Schritt, die für jeden Text $x \in \{0,1\}^*$ einen nichtleeren Bitstring $y(x) \in \bigcup_{n \geq 1} \{0,1\}^{tn}$ liefert
- Berechnung von $h_k(x)$:

```
1   y := y(x) = y_1 ... y_n, n ≥ 1, y_i ∈ {0,1}^t
2   z_0 := IV
3   for i = 1 to n do
4     z_i := E(k, z_{i-1} ⊕ y_i)
5   output h_k(x) = z_n
```

- Die MAC-Wert-Länge beträgt also t Bit

CBC-MACs

- Wird auf den Preprocessing-Schritt verzichtet, so lässt sich leicht ein Angriff mit 2 adaptiven Fragen ausführen
- Kennt der Angreifer die MAC-Werte $z = h_k(x)$ und $z' = h_k(x')$ für die Texte $x = x_1 \cdots x_n$ und $x' = (x_{n+1} \oplus IV \oplus z)x_{n+2} \cdots x_{n+m}$, wobei $|x_i| = t$ für $i = 1, \ldots, n+m$ ist, so muss auch der Text $x'' = x_1 \cdots x_{n+m}$ den MAC-Wert $h_k(x'') = z'$ haben
- Diesen Angriff kann man zwar ausschließen, indem man eine feste Länge nt für die Texte vorschreibt, wodurch die Anwendbarkeit des CBC-MACs allerdings einschränkt wird
- Der folgende Geburtstagsangriff ist aber auch bei fester Textlänge möglich

Geburtstagsangriff auf einen CBC-MAC

- Dieser Angriff ermöglicht es, mit $q+1$ MAC-Fragen (wobei $q \approx 1{,}17 \cdot 2^{\frac{t}{2}}$) den MAC-Wert $h_k(x)$ für einen zuvor nicht erfragten Text x zu finden, wobei $x = x_1 \ldots x_n \in \{0,1\}^{tn}$ abgesehen vom ersten t-Bitblock $x_1 \in \{0,1\}^t$ beliebig wählbar ist
- Hierzu wählt der Angreifer zunächst
  - $n-2$ beliebige Blöcke $x_3, \ldots, x_n \in \{0,1\}^t$ und
  - $q \approx 1{,}17 \cdot 2^{\frac{t}{2}}$ paarweise verschiedene Blöcke $x_1^1, \ldots, x_1^q \in \{0,1\}^t$
- Anschließend wählt er zufällig
  - q weitere Blöcke $x_2^1, \ldots, x_2^q \in \{0,1\}^t$ und

  erfragt die MAC-Werte $z_i = h_k(x^i)$ für die Texte $x^i = x_1^i x_2^i x_3 \cdots x_n$, $i = 1, \ldots, q$
- Wegen $x_1^i \neq x_1^j$ für $i \neq j$ sind auch die Texte $x^1, \ldots, x^q$ paarweise verschieden

Geburtstagsangriff auf einen CBC-MAC

- Seien $z_1^1, \ldots, z_1^q$ die nach der ersten Iteration des CBC-MACs berechneten Blöcke $z_1^i = E_k(IV \oplus x_1^i)$
- Da die Blöcke x_2^i zufällig gewählt werden, sind auch die Eingangsblöcke $z_1^i \oplus x_2^i$ für die zweite Iteration zufällig
- Es gilt also

$$\Pr[\exists i \neq j : z_1^i \oplus x_2^i = z_1^j \oplus x_2^j] = \Pr[\exists i \neq j : x_2^i = x_2^j] \approx \frac{1}{2}$$

- Die Gleichheit der Eingangsblöcke $z_1^i \oplus x_2^i$ und $z_1^j \oplus x_2^j$ für die zweite Iteration ist mit der Gleichheit der Ausgangsblöcke z_n^i und z_n^j der n-ten Iteration und damit mit der Gleichheit der zugehörigen MAC-Werte z^i und z^j äquivalent
- Daher kann der Angreifer das Indexpaar (i, j) mit $z_1^i \oplus x_2^i = z_1^j \oplus x_2^j$ auch leicht finden, sofern es existiert (was wir im Folgenden annehmen)

Geburtstagsangriff auf einen CBC-MAC

- Da $x_1^i \neq x_1^j$ gilt, sind auch die Blöcke $z_1^i = E_k(IV \oplus x_1^i)$ und $z_1^j = E_k(IV \oplus x_1^j)$ verschieden
- Wegen $z_1^i \oplus x_2^i = z_1^j \oplus x_2^j$ sind dann auch die beiden Blöcke x_2^i und x_2^j verschieden
- O.B.d.A. gelte $x_2^i \neq x_2$ (sonst vertauschen wir die Indizes i und j)
- Nun erfragt der Angreifer für $u = x_2^i \oplus x_2 \in \{0,1\}^t - \{0^t\}$ den MAC-Wert $\tilde{z}_j = h_k(\tilde{x}^j)$ für den Text $\tilde{x}^j = x_1^j(x_2^j \oplus u)x_3 \cdots x_n$, welcher zugleich MAC-Wert des Textes $\tilde{x}^i = x_1^i(x_2^i \oplus u)x_3 \cdots x_n = x_1^i x_2 x_3 \cdots x_n$ ist, den er zuvor nicht erfragt hat

Geburtstagsangriff auf einen CBC-MAC

Definition

- Sei $0 \leq \varepsilon \leq 1$ und sei $q \in \mathbb{N}$
- Ein (ε, q)-***Fälscher*** für einen MAC $\mathcal{H}$ ist ein probabilistischer Algorithmus $\mathcal{A}$, der q Fragen $x_1, \ldots, x_q$ stellt und aus den Antworten $z_i = h_k(x_i)$ mit Wahrscheinlichkeit mindestens ε (bei zufällig gewähltem Schlüssel k) ein Paar (x, z) mit $h_k(x) = z$ und $x \notin \{x_1, \ldots, x_q\}$ ausgibt

- Wir unterscheiden zwischen adaptiven und nicht-adaptiven Fragen
- Im ersten Fall darf der Text x_i von den MAC-Werten der Texte $x_1, \ldots, x_{i-1}$ abhängen, im zweiten nicht
- Zudem unterscheiden wir zwischen selektiven und existientiellen Fälschungen
- Im ersten Fall kann der Angreifer den MAC-Wert für einen Text x seiner Wahl generieren, im zweiten hat er keinen Einfluss auf die Wahl von x

Geburtstagsangriff auf einen CBC-MAC

Beispiel

- Der oben beschriebene Geburtstagsangriff auf einen CBC-MAC führt auf einen $(\frac{1}{2}, q+1)$-Fälscher für $q \approx 1{,}17 \cdot 2^{\frac{t}{2}}$
- Dabei ist nur die letzte MAC-Frage adaptiv und der Text x kann abgesehen vom ersten t-Bitblock beliebig vorgegeben werden ◁

- Eine Variante dieses Angriffs ist auch bei Verwendung einer Preprocessing-Funktion möglich
- Meist wird hierzu die Funktion $y : x \mapsto y(x) = y_0 \dots y_n$ mit $y_0 = bin_t(|x|)$ und $y_1 \dots y_n = x0^{nt-|x|}$ verwendet, wobei $n = \lceil |x|/t \rceil$ ist
- Der erste Block $y_0 = bin_t(|x|)$ kodiert also die Länge von x als Binärzahl, die mit führenden Nullen auf die Länge t erweitert wird, und der letzte Block wird ebenfalls mit Nullen auf die Länge t aufgefüllt

Kombination von Hashfunktion und MAC (HMAC)

Falls der Textraum eines MAC den Werteraum eines anderen MAC enthält, lassen sich diese leicht komponieren (Nested-MAC oder NMAC)

Definition

- Seien $\mathcal{H}_1 = (X, Y, K_1, F)$ mit $F = \{f_k \mid k \in K_1\}$ und $\mathcal{H}_2 = (Y, Z, K_2, G)$ mit $G = \{g_k \mid k \in K_2\}$ MACs
- Dann ist $\mathcal{H}_1 \circ \mathcal{H}_2 = (X, Z, K, H)$ die Komposition von $\mathcal{H}_1$ und $\mathcal{H}_2$, wobei $K = K_1 \times K_2$ und $H = \{g_{k_2} \circ f_{k_1} \mid (k_1, k_2) \in K\}$ ist

Beispiel

Wählt man für $\mathcal{H}_2$ einen MAC mit fester Textlänge und für $\mathcal{H}_1$ eine (schlüssellose) Hashfunktion (etwa `SHA-1`), so erhält man einen so genannten HMAC (Hash-MAC) ◁

Kombination von Hashfunktion und MAC (HMAC)

- Eine Variante hiervon ist der auf `SHA-1` basierende HMAC
- Hierbei werden zwei Varianten von `SHA-1` mit symmetrischen Schlüsseln komponiert, wobei jedoch beidesmal derselbe Schlüssel benutzt wird
- Seien

  $$ipad = \underbrace{36\ldots36}_{64mal} \text{ und } opad = \underbrace{5C\ldots5C}_{64mal}$$

  512 Bit Konstanten
- Dann berechnet sich HMAC wie folgt:

  $$\text{HMAC}_k(x) = \texttt{SHA-1}((k \oplus opad)\texttt{SHA-1}((k \oplus ipad)x))$$

- Hierbei fungiert die Funktion $f_k(x) = \texttt{SHA-1}((k \oplus ipad)x)$ als Hashfunktion mit Schlüssel, die beliebig lange Texte hasht, und der MAC $g_k(y) = \texttt{SHA-1}((k \oplus opad)y)$ wird nur auf Bitstrings der Länge 512+160=672 angewendet

Kombination von Hashfunktion und MAC (HMAC)

Wie der nächste Satz zeigt, genügt es, wenn f_k kollisionsresistent und g_k berechnungsresistent ist, um einen berechnungsresistenten HMAC mit beliebiger Eingabelänge zu erhalten

Definition

- Ein (ε, q)-***Kollisionsangreifer*** für einen MAC $\mathcal{H} = (X, Y, K, H)$ ist ein probabilistischer Algorithmus $\mathcal{A}$, der q Fragen $x_1, \ldots, x_n$ stellt und aus den Antworten $y_i = h_k(x_i)$ mit Wahrscheinlichkeit mindestens ε ein Paar (x, x') berechnet mit $h_k(x) = h_k(x')$
- Hierbei ist k der dem Angreifer unbekannte (und zufällig gewählte) Schlüssel

- Da der Angreifer den Schlüssel k nicht kennt, ist ein Kollisionsangriff gegen einen MAC $\mathcal{H}$ meist schwieriger zu realisieren als ein Kollisionsangriff gegen eine schlüssellose Hashfunktion
- Andererseits bringt die Kenntnis des Schlüssels bei einem Geburtstagsangriff keinen Vorteil

Kombination von Hashfunktion und MAC (HMAC)

Satz

- Seien $\mathcal{H}_1 = (X, Y, K_1, F)$ und $\mathcal{H}_2 = (Y, Z, K_2, G)$ MACs
- Falls für $\mathcal{H}_1$ kein adaptiver $(\varepsilon_1, q+1)$-Kollisionsangriff und für $\mathcal{H}_2$ kein adaptiver (ε_2, q)-Fälscher existieren, dann existiert für $\mathcal{H} = \mathcal{H}_1 \circ \mathcal{H}_2$ kein adaptiver $(\varepsilon_1 + \varepsilon_2, q)$-Fälscher

Beweis.

- Sei A ein adaptiver (ε, q)-Fälscher für $\mathcal{H}$
- Seien $x_1, \ldots, x_q$ die Fragen, die A an sein Orakel $g_{k_2} \circ f_{k_1}$ stellt, und seien $z_i = g_{k_2}(f_{k_1}(x_i))$ die erhaltenen Antworten
- Zudem sei (x, z) die Ausgabe von A
- Wir müssen zeigen, dass die Erfolgswk von A

$$\Pr[\underbrace{x \notin \{x_1, \ldots, x_q\}}_{B} \wedge \underbrace{g_{k_2}(f_{k_1}(x)) = z}_{C}] < \varepsilon_1 + \varepsilon_2$$

  ist, wobei (k_1, k_2) zufällig aus $K = K_1 \times K_2$ gewählt wird
- Hierzu zeigen wir zwei Behauptungen

Kombination von Hashfunktion und MAC (HMAC)

Behauptung 1

$$\Pr[\underbrace{x \notin \{x_1, \ldots, x_q\}}_{B} \wedge \underbrace{f_{k_1}(x) \in \{f_{k_1}(x_1), \ldots, f_{k_1}(x_q)\}}_{D}] < \varepsilon_1$$

- Zum Beweis der Behauptung betrachten wir den adaptiven Kollisionsangreifer A' gegen $\mathcal{H}_1$, der zufällig einen Schlüssel $k_2 \in K_2$ wählt und A wie folgt simuliert:

  Für jede Frage x_i von A erfragt A' den MAC-Wert $y_i = f_{k_1}(x_i)$ und gibt an A die Antwort $z_i = g_{k_2}(y_i)$ zurück. Sobald A ein Paar (x, z) ausgibt, erfragt A' den MAC-Wert $y = f_{k_1}(x)$ und gibt im Fall $x \notin \{x_1, \ldots, x_q\} \wedge y \in \{y_1, \ldots, y_q\}$ das Paar (x, x_i) für einen beliebigen Index i mit $y = y_i$ aus

- Da A' genau dann Erfolg hat, wenn das Ereignis $B \cap D$ eintritt, folgt die Behauptung
- Als nächstes zeigen wir folgende Behauptung 2

Kombination von Hashfunktion und MAC (HMAC)

Behauptung 2

$$\Pr[\underbrace{f_{k_1}(x) \notin \{f_{k_1}(x_1), \ldots, f_{k_1}(x_q)\}}_{\overline{D}} \wedge \underbrace{g_{k_2}(f_{k_1}(x)) = z}_{C}] < \varepsilon_2$$

- Zum Beweis der Behauptung betrachten wir den adaptiven Fälscher A'' gegen $\mathcal{H}_2$, der zufällig einen Schlüssel $k_1 \in K_1$ wählt und A wie folgt simuliert:

  > *A'' gibt bei jeder Anfrage x_i von A die Antwort des Orakels g_{k_2} auf die Frage $y_i = f_{k_1}(x_i)$ zurück und sobald A ein Paar (x, z) ausgibt, gibt A'' das Paar $(f_{k_1}(x), z)$ aus*

- Da A'' genau dann Erfolg hat, wenn das Ereignis $\overline{D} \cap C$ eintritt, folgt die Behauptung

Kombination von Hashfunktion und MAC (HMAC)

Satz

- Seien $\mathcal{H}_1 = (X, Y, K_1, F)$, $\mathcal{H}_2 = (Y, Z, K_2, G)$ MACs
- Falls für $\mathcal{H}_1$ kein adaptiver $(\varepsilon_1, q+1)$-Kollisionsangriff und für $\mathcal{H}_2$ kein adaptiver (ε_2, q)-Fälscher existieren, dann existiert auch für $\mathcal{H} = \mathcal{H}_1 \circ \mathcal{H}_2$ kein adaptiver $(\varepsilon_1 + \varepsilon_2, q)$-Fälscher

Beweis (Ende).

- Nun folgt für die Erfolgswk von A

$$\Pr[\underbrace{x \notin \{x_1, \ldots, x_q\}}_{B} \wedge \underbrace{g_{k_2}(f_{k_1}(x)) = z}_{C}]$$

$$= \underbrace{\Pr(B \cap D \cap C)}_{<\varepsilon_1} + \underbrace{\Pr(B \cap \overline{D} \cap C)}_{<\varepsilon_2} < \varepsilon_1 + \varepsilon_2$$

□

Elliptische Kurven über den reellen Zahlen

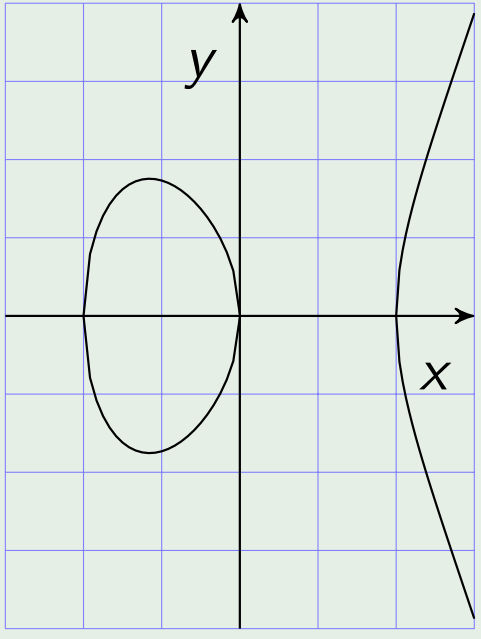
Definition

- Seien $a, b \in \mathbb{R}$
- Eine elliptische Kurve E über $\mathbb{R}$ enthält alle Lösungen $(x, y) \in \mathbb{R}^2$ der Gleichung $y^2 = x^3 + ax + b$ und zusätzlich den Punkt $\mathcal{O}$ (Punkt im Unendlichen; siehe Übungen)
- Im Fall $4a^3 + 27b^2 = 0$ heißt E ***singulär***, sonst ***nicht-singulär***

Elliptische Kurven über den reellen Zahlen

Beispiel

- Betrachte die durch $y^2 = x^3 - 4x$ definierte elliptische Kurve E:



Punkte: $(-2, 0)$, $(0, 0)$, $(2, 0)$, $(-1, \sqrt{3})$, $(-1, -\sqrt{3})$, $(3, \sqrt{15})$, $(3, -\sqrt{15})$

Elliptische Kurven über den reellen Zahlen

- Auf den nicht-singulären Punkten von E lässt sich eine additive Gruppenoperation $+$ definieren
- Die Idee dabei ist, dass die Addition von 3 beliebigen Punkten von E, die auf einer Geraden liegen, das neutrale Element $\mathcal{O}$ ergeben soll
- Hierbei werden Tangentialpunkte doppelt und Wendepunkte dreifach gezählt und den parallel zur y-Achse verlaufenden Geraden wird zusätzlich noch der Punkt $\mathcal{O}$ hinzugerechnet
- D.h. alle Geraden, die parallel zur y-Achse verlaufen, schneiden sich im Punkt $\mathcal{O}$ und es werden nur solche Geraden g betrachtet, auf denen bei dieser Zählweise 3 Punkte von E liegen

Berechnung von $P + Q$

- Um nun die Summe $R = P + Q$ von zwei gegebenen Punkten $P = \{x_1, y_1\}$ und $Q = \{x_2, y_2\}$ zu berechnen, bestimmen wir zuerst die Gerade g, auf denen P und Q liegen, wobei g im Fall $P = Q$ die Tangente an E im Punkt P ist
- Falls g parallel zur y-Achse verläuft, ist $x_1 = x_2$ und $y_1 = -y_2$ (also $Q = (x_1, -y_1)$)
- Da in diesem Fall zudem der Punkt $\mathcal{O}$ auf g liegt, erhalten wir die Gleichung $P + Q(+\mathcal{O}) = \mathcal{O}$ bzw. $-P = Q = (x_1, -y_1)$

Berechnung von $P + Q$

Falls g nicht parallel zur y-Achse verläuft, können wir $P + Q$ wie folgt berechnen

- Im Fall $P \neq Q$ gilt $x_1 \neq x_2$
- Zudem ist $g = \{(x, y) \in \mathbb{R}^2 | y = \lambda x + \mu\}$ mit $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$ und $\mu = y_1 - \lambda x_1 = y_2 - \lambda x_2$
- Wir zeigen zuerst, dass es einen Punkt $R = (x_3, y_3) \in \mathbb{R}^2$ gibt mit

$$E \cap g = \{P, Q, R\}$$

- Für alle $(x, y) \in E \cap g$ gilt

$$(\lambda x + \mu)^2 = x^3 + ax + b$$

$$\rightsquigarrow \underbrace{x^3 - \lambda^2 x^2 + (a - 2\mu\lambda)x + b - \mu^2}_{p(x)} = 0$$

- p lässt sich in $\mathbb{C}$ vollständig in Linearfaktoren zerlegen:

$$p(x) = (x - x_1)(x - x_2)(x - x_3)$$

Berechnung von $P + Q$ im Fall $P \neq Q$

- p lässt sich in $\mathbb{C}$ vollständig in Linearfaktoren zerlegen:

  $$p(x) = (x - x_1)(x - x_2)(x - x_3)$$

- Da sich der Koeffizient $-\lambda^2$ von x^2 aus der linearen Zerlegung von $p(x)$ zu

  $$-\lambda^2 = -x_1 - x_2 - x_3$$

  berechnet, muss $x_3 = \lambda^2 - x_1 - x_2$ sein
- Da R auch auf g liegt, ist zudem $y_3 = \lambda(x_3 - x_1) + y_1$
- Folglich ist $P + Q = -R = (x_3, -y_3) = (\lambda^2 - x_1 - x_2, \lambda(x_1 - x_3) - y_1)$

Berechnung von $P + Q$ im Fall $P = Q$

- Im Fall $P = Q$ gilt $x_1 = x_2$ und $y_1 = y_2 \neq 0$
- Sei g die Tangente durch P an E
- Wir zeigen, dass es einen Punkt $R = (x_3, y_3) \in \mathbb{R}^2$ gibt mit

$$g \cap E = \{P, R\}$$

- Die Steigung λ von g erhalten wir durch implizites Differenzieren:

$$\lambda = \frac{-\frac{\partial F}{\partial x}(x_1, y_1)}{\frac{\partial F}{\partial y}(x_1, y_1)} = \frac{3x_1^2 + a}{2y_1},$$

wobei $F(x, y) = y^2 - x^3 - ax - b$ ist

- Zur Begründung sei

$$T(x, y) = c(x - x_1) + d(y - y_1)$$

die Tangentialebene an die Fläche $F(x, y)$ im Punkt $(x_1, y_1, F(x_1, y_1)) = (x_1, y_1, 0)$

Berechnung von $P + Q$ im Fall $P = Q$

- Zur Begründung sei

$$T(x, y) = c(x - x_1) + d(y - y_1)$$

die Tangentialebene an die Fläche $F(x, y)$ im Punkt $(x_1, y_1, F(x_1, y_1)) = (x_1, y_1, 0)$

- Dann gilt

$$c = \frac{\partial F}{\partial x}(x_1, y_1) = -3x_1^2 - a$$

und

$$d = \frac{\partial F}{\partial y}(x_1, y_1) = 2y_1$$

- Da die Tangente g sowohl in der Tangentialebene T als auch in der x, y-Ebene verläuft, folgt

$$\begin{aligned} (x, y) \in g \quad &\Leftrightarrow \quad T(x, y) = 0 \\ &\Leftrightarrow \quad y - y_1 = -\frac{c}{d}(x - x_1), \end{aligned}$$

woraus sich $\lambda = -\frac{c}{d}$ ergibt

Berechnung von $P + Q$ im Fall $P = Q$

- Genau wie im Fall **P** $\neq$ **Q** erhalten wir nun
  $P+Q = P+P = 2P = -R = (x_3, -y_3)) = (\lambda^2 - x_1 - x_2, \lambda(x_1 - x_3) - y_1)$
  mit $\lambda = \frac{3x_1^2+a}{2y_1}$

Satz

E bildet mit $\mathcal{O}$ als neutralem Element und $+$ als Addition eine abelsche Gruppe, d.h.

- $+$ ist abgeschlossen auf E
- $+$ ist kommutativ
- Jeder Punkt hat ein Inverses $-P$
- P ist selbstinvers, falls $P = -P$ ist;
  dies gilt für $P = \mathcal{O}$ und alle Kurvenpunkte der Form $P = (x, 0)$
- $+$ ist assoziativ (ohne Beweis!)

Elliptische Kurven über endlichen Körpern

Definition

- Sei $\mathbb{F}_q$ ein endlicher Körper mit $q = p^n$ für eine Primzahl $p > 3$ (der Fall $p = 2$ wird in den Übungen betrachtet)
- Für $a, b \in \mathbb{F}_q$ mit $4a^3 + 27b^2 \neq 0$ heißt

$$E = \{(x, y) \in \mathbb{F}_q \mid y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\}$$

  elliptische Kurve über $\mathbb{F}_q$
- Die Gruppenoperation $+$ ist auf E wie folgt definiert:
  - $\mathcal{O}$ ist neutrales Element, d.h. $\forall P \in E - \{\mathcal{O}\} : P + \mathcal{O} = \mathcal{O} + P = P$
  - Das Inverse zu $P = (x, y) \in E \setminus \{\mathcal{O}\}$ ist $-P = \overline{P} = (x, -y)$
  - Für $P, Q \in E \setminus \{\mathcal{O}\}$ ist

$$P + Q = \begin{cases} \mathcal{O}, & P = \overline{Q} \\ R, & \textit{sonst} \end{cases}$$

Definition

- Die Gruppenoperation $+$ ist auf E wie folgt definiert:
  - $\mathcal{O}$ ist neutrales Element, d.h. $\forall P \in E - \{\mathcal{O}\} : P + \mathcal{O} = \mathcal{O} + P = P$
  - Das Inverse zu $P = (x, y) \in E \setminus \{\mathcal{O}\}$ ist $-P = \overline{P} = (x, -y)$
  - Für $P, Q \in E \setminus \{\mathcal{O}\}$ ist

$$P + Q = \begin{cases} \mathcal{O}, & P = \overline{Q} \\ R, & \textit{sonst} \end{cases}$$

- Dabei wird $R = (x_3, y_3)$ wie folgt aus $P = (x_1, y_1)$ und $Q = (x_2, y_2)$ berechnet:

$$\begin{aligned} x_3 &= \lambda^2 - x_1 - x_2 \\ y_3 &= \lambda(x_1 - x_3) - y_1 \end{aligned} \qquad \text{mit } \lambda = \begin{cases} (y_2 - y_1)(x_2 - x_1)^{-1}, & P \neq Q \\ (3x_1^2 + a)(2y_1)^{-1}, & P = Q \end{cases}$$

Elliptische Kurven über endlichen Körpern

Satz

$(E, \mathcal{O}, +)$ bildet eine abelsche Gruppe (ohne Beweis)

Beispiel

- Sei E definiert durch $y^2 = x^3 + x + 6$ über $\mathbb{Z}_p$, $p = 11$
- Im Fall $p \equiv_4 3$ lassen sich die Wurzeln y von quadratischen Resten $z \in QR_p = \{y^2 \bmod p \mid y \in \mathbb{Z}_p^*\}$ durch $\pm z^{\frac{p+1}{4}} \bmod p$ bestimmen

x	0	1	2	3	4	5	6	7	8	9	10
$z = x^3 + x + 6$	6	8	5	3	8	4	8	4	9	7	4
$y = \pm\sqrt{z} \bmod 11$	–	–	4; 7	5; 6	–	2; 9	–	2; 9	3; 8	–	2; 9

- Da die Gruppe $(E, +, \mathcal{O})$ die Größe $\|E\| = 13$ hat und 13 prim ist, ist die Ordnung jedes Elements der Kurve 1 oder 13
- Da nur das neutrale Element $\mathcal{O}$ die Ordnung 1 haben kann, haben alle anderen Elemente die Ordnung 13 und sind Erzeuger der Gruppe
- Folglich ist $(E, +, \mathcal{O})$ zyklisch und somit isomorph zu $(\mathbb{Z}_{13}, +, 0)$

Elliptische Kurven über endlichen Körpern

Beispiel (Fortsetzung)

- Untenstehende Tabelle zeigt die Vielfachen des Punktes $P = (2, 7) \in E$
- Berechnung von $2P = (2, 7) + (2, 7) = (5, 2)$:

$$\begin{aligned} \lambda &= (3 \cdot 2^2 + 1)(2 \cdot 7)^{-1} \bmod 11 = 2 \cdot 3^{-1} = 2 \cdot 4 \bmod 11 = 8 \\ x_3 &= 8^2 - 2 - 2 \bmod 11 = 5 \\ y_3 &= 8(2 - 5) - 7 \bmod 11 = 2 \end{aligned}$$

- Berechnung von $3P = 2P + P = (5, 2) + (2, 7) = (8, 3)$:

$$\begin{aligned} \lambda &= (7 - 2)(2 - 5)^{-1} \bmod 11 = 5 \cdot (-3)^{-1} \bmod 11 = 2 \\ x_3 &= 2^2 - 5 - 2 \bmod 11 = 8 \\ y_3 &= 2 \cdot (5 - 8) - 2 \bmod 11 = 3 \end{aligned}$$

m	1	2	3	4	5	6	7	8	9	10	11	12	13
mP	(2,7)	(5,2)	(8,3)	(10,2)	(3,6)	(7,9)	(7,2)	(3,5)	(10,9)	(8,8)	(5,9)	(2,4)	$\mathcal{O}$

Elliptische Kurven über endlichen Körpern

Satz (Hasse)

Für die Anzahl $\|E\|$ von Punkten einer elliptischen Kurve über einem endlichen Körper $\mathbb{F}_q$ gilt

$$q + 1 - 2\sqrt{q} \leq \|E\| \leq q + 1 + 2\sqrt{q} \quad \text{(ohne Beweis)}$$

Bemerkung

Es gibt einen effizienten Algorithmus (von Schoof) mit Zeitkomplexität $O(\log^8 q)$, der $\|E\|$ bei Eingabe von a, b und q berechnet

Satz

Sei E eine elliptische Kurve über $\mathbb{F}_q$. Dann ist $(E, \mathcal{O}, +)$ isomorph zu $(\mathbb{Z}_{n_1}, 0, +) \times (\mathbb{Z}_{n_2}, 0, +)$, wobei $n_1, n_2 \in \mathbb{N}^+$ sind und n_1 Teiler von n_2 und von $q - 1$ ist (ohne Beweis)

Elliptische Kurven über endlichen Körpern

Satz

Sei E eine elliptische Kurve über $\mathbb{F}_q$. Dann ist $(E, \mathcal{O}, +)$ isomorph zu $(\mathbb{Z}_{n_1}, 0, +) \times (\mathbb{Z}_{n_2}, 0, +)$, wobei $n_1, n_2 \in \mathbb{N}^+$ sind und n_1 Teiler von n_2 und von $q - 1$ ist (ohne Beweis)

Bemerkung

- Falls n_1 ein Teiler von n_2 ist, ist die (additive) Gruppe $\mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2}$ genau dann zyklisch, wenn $n_1 = 1$ (und somit $\mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2} \cong \mathbb{Z}_{n_2}$) ist
- Eine hinreichende Bedingung hierfür ist, dass $\|E\|$ quadratfrei (also das Produkt von paarweise verschiedenen Primzahlen) ist
- Im Fall $n_1 > 1$ ist E dagegen nicht zyklisch, hat aber eine nicht-triviale zyklische Untergruppe, die zu $\mathbb{Z}_{n_2}$ isomorph ist und für kryptografische Anwendungen benutzt werden kann

Kompakte Darstellung von Punkten auf E

Bemerkung

- Für den Fall, dass sich Quadratwurzeln effizient in $\mathbb{F}_q$ berechnen lassen, gibt es eine einfache Möglichkeit, Punkte auf einer elliptischen Kurve über $\mathbb{F}_q$ kompakter darzustellen
- Ist zum Beispiel $q = p$ prim mit $p \equiv_4 3$, so lassen sich die Wurzeln $\pm\sqrt{z} \bmod p$ von $z \in QR_p = \{x^2 \bmod p \mid x \in \mathbb{Z}_p^*\}$ effizient mittels $\pm z^{(p+1)/4} \bmod p$ berechnen
- Folgende Funktion liefert dann eine kompakte Darstellung: `PointCompress`: $E - \{\mathcal{O}\} \to \mathbb{Z}_p \times \mathbb{Z}_2$ mit $(x, y) \mapsto (x, y \bmod 2)$

Kompakte Darstellung von Punkten auf E

- Für die Rekonstruktion können wir folgende Prozedur benutzen
- Sei E eine elliptische Kurve $y^2 = x^3 + ax + b$ über $\mathbb{F}_p$ und sei $p(x) = x^3 + ax + b$

Prozedur `PointDeCompress`(x, b)

1 $z := p(x) \bmod p$
2 $y := z^{(p+1)/4} \bmod p$
3 **if** $y^2 \equiv_p z$ **then**
4 **if** $y \not\equiv_2 b$ **then** $y := p - y$
5 **output**(x, y)
6 **else** **output**(“ error ”)

Berechnung von Vielfachen von Punkten auf E

- In $\mathbb{Z}_m^*$ berechnen wir Potenzen a^e mod m durch 'wiederholtes Quadieren und Multiplizieren'
- Ähnlich können wir in einer elliptischen Kurve E die Vielfachen mP eines Punktes P durch 'wiederholtes Verdoppeln und Addieren' berechnen
- Da in E additive Inverse sehr leicht zu berechnen sind, kann mP durch 'wiederholtes Verdoppeln, Addieren und Subtrahieren' noch effizienter berechnet werden
- Hierzu repräsentieren wir m in NAF-Darstellung (Non Adjacent Form)

Berechnung von Vielfachen von Punkten auf E

Definition

- Eine Folge $(c_{l-1}, \ldots, c_0) \in \{-1, 0, 1\}^l$ heißt ***SBR-Darstellung*** (Signed Binary Representation) einer Zahl $c \in \mathbb{Z}$, falls

$$\sum_{i=0}^{l-1} c_i 2^i = c$$

  ist
- Ist von je zwei benachbarten Ziffern c_i mindestens eine 0, so heißt $(c_{l-1}, \ldots, c_0)$ ***NAF-Darstellung*** (Non-Adjacent Form) von c

Berechnung von Vielfachen von Punkten auf E

Beispiel

Sowohl $(0, 1, 0, 1, 1)$ als auch $(1, 0, -1, 0, -1)$ sind SBR-Darstellungen von $c = 1 + 2 + 8 = 11 = -1 - 4 + 16$ ◁

Satz

Jede Zahl $c \in \mathbb{Z}$ hat eine eindeutige NAF-Darstellung
(Beweis siehe Übungen)

Berechnung von Vielfachen von Punkten auf E

Berechnung einer NAF-Darstellung aus der Binärdarstellung

- Ersetze jeden Teilstring der Form $(0, 1, \ldots, 1)$ von rechts beginnend durch den Teilstring $(1, 0, \ldots, 0, -1)$
- Um z.B. die NAF-Darstellung von $c = 47$ zu berechnen, bestimmen wir zuerst die Binärdarstellung von c
- Es gilt $47_{10} = 101111_2$
- Mit obiger Transformationsregel ergibt sich

$$
\begin{aligned}
& (0,\ 1,\ \underbrace{0,\ 1,\ 1,\ 1,\ 1}) \\
\rightsquigarrow\ & (\underbrace{0,\ 1,\ 1}, 0,\ 0,\ 0, -1) \\
\rightsquigarrow\ & (1,\ 0, -1,\ 0,\ 0,\ 0, -1)
\end{aligned}
$$

Berechnung von Vielfachen von Punkten auf E

- Sei $(c_s, \ldots, c_0)$ eine SBR-Darstellung einer Zahl $c \in \mathbb{Z}$
- Zur effizienten Berechnung von $Q = cP$ benutzen wir das Horner-Schema

$$c = \sum_{j=0}^{s} c_j 2^j = (\ldots \underbrace{(\ldots (c_s 2 + c_{s-1})2 + \cdots + c_i)}_{d_i} 2 + \cdots + c_1)2 + c_0$$

- Dieses führt auf das folgende iterative Schema zur Berechnung der Punkte $Q_i = d_i P = \sum_{j=i}^{s} c_j 2^{j-i} P$:

$$Q_i = \begin{cases} \mathcal{O}, & i = s+1 \\ 2Q_{i+1} + c_i P, & i = s, \ldots, 0 \end{cases}$$

Berechnung von Vielfachen von Punkten auf E

- Damit erhalten wir folgenden Algorithmus zur Berechnung von $Q = Q_0 = cP$:

Prozedur `DoubleAddSub`$(P, c_s, \ldots, c_0)$

1 $Q := \mathcal{O}$
2 **for** $i := s$ **downto** 0 **do**
3 $\quad Q := 2Q + c_i P$
4 **output**(Q)

- Da eine $(s+1)$-Bitzahl im Durchschnitt $s/2$ Nullen in Binärdarstellung und $(2/3)s$ Nullen in NAF-Darstellung enthält (siehe Übungen), benötigt DoubleAddSub bei Verwendung von NAF ca. $(4/3)s$ Additionen/Subtraktionen im Vergleich zu ca. $(3/2)s$ Additionen im Binärfall
- Dies entspricht einer Beschleunigung um ca. 11 Prozent

Digitale Signaturverfahren

Eigenschaften von handschriftlichen Signaturen

- Die durch die Unterschrift gekennzeichnete Person hat überprüfbar die Unterschrift geleistet
- Die Unterschrift ist nicht auf ein anderes Dokument übertragbar, ohne ihre Gültigkeit zu verlieren
- Das signierte Dokument kann nachträglich nicht unbemerkt verändert werden

Eine direkte Übertragung dieser Eigenschaften in die digitale Welt ist nicht möglich

Lösung:

Die digitale Signatur wird nicht physikalisch, sondern logisch (inhaltlich) an ein elektronisches Dokument bzw. Text gebunden und die Fähigkeit, einen individuellen Schriftzug auszuführen, wird durch geheimes Wissen ersetzt

Digitale Signaturverfahren

Definition

Ein ***digitales Signaturverfahren*** besteht aus

- einer Menge X von Texten
- einer endlichen Menge Y von Signaturen
- einem Schlüsselraum K
- einer Menge $S \subseteq K \times K$ von Schlüsselpaaren $(\hat{k}, k)$, bestehend aus einem Signierschlüssel $\hat{k}$ und einem Verifikationsschlüssel k
- einem Signieralgorithmus $sig : K \times X \to Y$ und
- einem Verifikationsalgorithmus $ver : K \times X \times Y \to \{0, 1\}$, so dass $ver(k, x, y) = 1$ für alle Paare $(\hat{k}, k) \in S$ und $(x, y) \in X \times Y$ mit $y = sig(\hat{k}, x)$ gilt

Im Fall $ver(k, x, y) = 1$ heißt y gültige Signatur für den Text x (unter k), andernfalls ungültig

Digitale Signaturverfahren

- Ein wichtiger Unterschied zu MACs besteht darin, dass digitale Signaturverfahren asymmetrisch sind
- Aufgrund dieser Asymmetrie kann Bob nämlich auch einem Dritten gegenüber nachweisen, dass eine von Alice erzeugte Signatur y tatsächlich von Alice stammt
- Bei Verwendung eines MACs zur Authentifikation einer Nachricht x könnte Bob die Nachricht manipuliert und den MAC-Wert auch selbst erzeugt haben, weshalb Alice ihre Urheberschaft von x erfolgreich abstreiten kann
- Ein weiterer Vorteil von digitalen Signaturen gegenüber MACs ist, dass eine von Alice geleistete Signatur von allen verifizierbar ist, sofern sie den öffentlichen Verifikationsschlüssel von Alice kennen
- Um bspw. die Authentizität eines Software-Updates x zu gewährleisten, kann eine SW-Firma x zusammen mit ihrer Signatur y für x verschicken
- Bei Verwendung eines MACs müsste die SW-Firma dagegen mit jedem einzelnen Kunden K_i einen symmetrischen Schlüssel k_i vereinbaren und den zugehörigen MAC-Wert $y_i = h_{k_i}(x)$ versenden

Klassifikation von Angriffen gegen Signaturverfahren

Angriff bei bekanntem Verifikationsschlüssel (key-only attack)

Dem Angreifer ist nur der öffentliche Verifikationsschlüssel k bekannt und er versucht, ein Paar (x, y) mit $ver(k, x, y) = 1$ zu finden. Jedes solche Paar, das nicht von Alice unter Verwendung des geheimen Signierschlüssels erzeugt wurde, wird als Fälschung bezeichnet

Angriff bei bekannter Signatur (known signature attack)

Der Angreifer kennt neben k die Signaturen $y_i = sig(\hat{k}, x_i)$ für eine Reihe von Texten $x_1, \ldots, x_q$, auf deren Auswahl er keinen Einfluss hat, und versucht, eine Fälschung (x, y) mit $x \notin \{x_1, \ldots, x_q\}$ zu finden

Angriff bei frei wählbaren Texten (chosen document attack)

Der Angreifer kann die Texte $x_1, \ldots, x_q$ selbst wählen, erhält die Signaturen aber erst, nachdem er alle Texte vorgelegt hat

Angriff bei adaptiv wählbaren Texten

Der Angreifer kann die Wahl des Textes x_{i+1} von den Signaturen $y_1, \ldots, y_i$ abhängig machen

Erfolgskriterien für die Fälschung digitaler Signaturen

uneingeschränktes Fälschungsvermögen (total break)

Der Angreifer hat einen Weg gefunden, die Funktion $x \mapsto sig(\hat{k}, x)$ bei Kenntnis von k effizient zu berechnen

selektives Fälschungsvermögen (selective forgery)

Der Angreifer kann für Texte seiner Wahl die zugehörigen Signaturen bestimmen (eventuell mit Hilfe des legalen Unterzeichners)

nichtselektives (existentielles) Fälschungsvermögen

Der Angreifer kann für bestimmte Texte x, auf deren Wahl er keinen Einfluss hat, die zugehörige digitale Signatur bestimmen

Das RSA-Kryptosystem

- Das RSA-Kryptosystem wurde 1978 von Rivest, Shamir und Adleman veröffentlicht
- Während es beim Primzahlproblem nur um die Frage „Ist n prim?“ geht, muss beim Faktorisierungsproblem im Falle einer zusammengesetzten Zahl mindestens ein nicht-trivialer Faktor berechnet werden
- Genauer gesagt beruht das RSA-Verfahren darauf, dass die Primzahleigenschaft zwar effizient getestet werden kann, aber keine effizienten Faktorisierungsalgorithmen bekannt sind

Schlüsselgenerierung

Für jeden Teilnehmer X werden zwei Primzahlen p, q und zwei Exponenten e, d mit $ed \equiv_{\varphi(n)} 1$ generiert, wobei $n = pq$ und $\varphi(n) = (p-1)(q-1)$ ist

Öffentlicher Schlüssel: $k_X = (e, n)$

Privater Schlüssel: $k'_X = (d, n)$

Das RSA-Kryptosystem

Ver- und Entschlüsselung

- Jede Nachricht x wird durch eine Folge $x_1, x_2, \ldots$ von Zahlen $x_i \in \mathbb{Z}_n$ dargestellt, die einzeln wie folgt ver- und entschlüsselt werden:
  - $\mathrm{RSA}((e,n),x) = x^e \bmod n$
  - $\mathrm{RSA}^{-1}((d,n),y) = y^d \bmod n$
- Der Schlüsselraum ist also

  $$K = \{(c,n) \mid \text{es gibt Primzahlen } p \text{ und } q \text{ mit } n = pq \text{ und } c \in \mathbb{Z}^*_{\varphi(n)}\}$$

  und

  $$S = \{((e,n),(d,n)) \in K \times K \mid ed \equiv_{\varphi(n)} 1\}$$

  ist die Menge aller zueinander passenden Schlüsselpaare
- Die Chiffrierfunktionen $\mathrm{RSA}_{(e,n)}$ und $\mathrm{RSA}^{-1}_{(d,n)}$ sind durch Wiederholtes Quadrieren und Multiplizieren effizient berechenbar

Das RSA-Kryptosystem

Ver- und Entschlüsselung

Der folgende Satz garantiert die Korrektheit des RSA-Systems

Satz

Für jedes Schlüsselpaar $((e,n),(d,n)) \in S$ und alle $x \in \mathbb{Z}_n$ gilt

$$x^{ed} \equiv_n x$$

Beweis.

- Sei $n = pq$ und sei z eine natürliche Zahl mit $ed = z\varphi(n) + 1$
- Wir zeigen $x^{ed} \equiv_p x$. Die Kongruenz $x^{ed} \equiv_q x$ folgt analog und beide Kongruenzen zusammen implizieren $x^{ed} \equiv_n x$
- Wegen $\varphi(n) = (p-1)(q-1)$ und wegen $x^{p-1} \equiv_p 1$ für $x \not\equiv_p 0$ folgt

$$x^{ed} = x^{z\varphi(n)+1} = x^{z(p-1)(q-1)}x = \left(x^{p-1}\right)^{z(q-1)}x \equiv_p x$$

□

Das RSA-Signaturverfahren

Definition

- Beim ***RSA-Signaturverfahren*** ist

$$K = \{(a, n) | n = pq \text{ für Primzahlen } p, q \text{ und } a \in \mathbb{Z}^*_{\varphi(n)}\}$$

und S die Relation $S = \{((d, n), (e, n)) \in K \times K | de \equiv_{\varphi(n)} 1\}$

- Signiert wird mittels $sig(d, n, x) := x^d \bmod n$, wobei $X = Y = \mathbb{Z}_n$ ist
- Die Verifikationsbedingung ist

$$ver(e, n, x, y) = \begin{cases} 1, & y^e \equiv_n x \\ 0, & sonst \end{cases}$$

Satz

Für alle $((d, n), (e, n)) \in S$ und $x, y \in \mathbb{Z}_n$ gilt

$$ver(e, n, x, y) = \begin{cases} 1, & sig(d, n, x) = y, \\ 0, & sonst \end{cases}$$

Der Beweis folgt direkt aus der Korrektheit des RSA-Kryptosystems

Das RSA-Signaturverfahren

- Wir betrachten eine Reihe von Angriffen gegen das RSA-Signaturverfahren und überlegen anschließend, durch welche Maßnahmen sich diese abwehren lassen
- Ein Angreifer kann leicht eine existentielle Fälschung bei bekanntem Verifikationsschlüssel erhalten, indem er zu einer beliebigen Signatur $y \in Y$ den Text $x = y^e \bmod n$ wählt
- Zudem ist eine existentielle Fälschung bei bekannten Signaturen möglich, falls der Angreifer zwei signierte Texte $(x_1, y_1), (x_2, y_2)$ mit $ver(k, x_i, y_i) = 1$ kennt
- Wegen $y_i^e \equiv_n x_i$ für $i = 1, 2$ folgt nämlich $(y_1 y_2)^e \equiv_n y_1^e y_2^e \equiv_n x_1 x_2$ und somit $ver(k, x_1 x_2 \bmod n, y_1 y_2 \bmod n) = 1$
- Weiterhin ist eine selektive Fälschung bei frei wählbarem Text möglich
- Kennt der Angreifer nämlich bereits die Signatur y' für einen beliebigen Text $x' \in \mathbb{Z}_n^*$ und kann er sich die Signatur y'' für $x'' = xx'^{-1} \bmod n$ beschaffen, so kann er daraus die Signatur $y = y'y'' \bmod n$ für den Text x berechnen

Das RSA-Signaturverfahren

- Diese Angriffe kann man vereiteln, indem man den Text x mit Redundanz versieht (indem man z.B. anstelle von x den Text xx signiert)
- Um auch längere Texte effizient signieren zu können, wird i.a. jedoch eine geeignete Hashfunktion h benutzt und nicht der gesamte Text x, sondern nur der Hashwert $h(x)$ signiert

Das RSA-Signaturverfahren

Bei der Signaturerstellung benötigte Eigenschaften einer Hashfunktion h

- Die verwendete Hashfunktion h sollte die Einwegeigenschaft haben, da sonst der Angreifer zu einem $y \in Y$ einen passenden Text x mit $h(x) = y$ bestimmen kann (zumindest wenn das Signaturverfahren anfällig gegen eine existentielle Fälschung ist, wie etwa RSA)
- Angenommen der Angreifer kennt bereits ein Paar (x, y) mit $ver(k, h(x), y) = 1$
- Dann sollte h zumindest schwach kollisionsresistent sein, da sonst der Angreifer ein x' mit $h(x') = h(x)$ berechnen und das Paar (x', y) bestimmen könnte
- Falls sich der Angreifer für bestimmte von ihm selbst gewählte Texte x die zugehörige Signatur y beschaffen kann, so sollte h sogar kollisionsresistent sein
- Andernfalls könnte der Angreifer ein Kollisonspaar (x, x') für h finden, sich den (unverdächtigen) Text x signieren lassen und die erhaltene Signatur y für den Text x' verwenden

Diskrete Logarithmen

- Für ein beliebiges Element a einer multiplikativen Gruppe G ist die Exponentiation $\exp_{G,a} : x \mapsto a^x$ zur Basis a eine Bijektion zwischen der Menge $\mathbb{Z}_{ord(a)} = \{0, 1, \ldots, ord(a) - 1\}$ und der Untergruppe $\langle a \rangle$
- Die zugehörige Umkehrabbildung spielt in der Kryptografie eine wichtige Rolle

Definition

- Seien $a, b \in G$ mit $b \in \langle a \rangle$
- Dann heißt der eindeutig bestimmte Exponent $x \in \mathbb{Z}_{\text{ord}(a)}$ mit $a^x = b$ ***Index*** oder ***diskreter Logarithmus von b zur Basis a in G***, kurz

$$x = \log_{G,a}(b)$$

- Im Fall $G = \mathbb{Z}_m^*$ schreiben wir auch einfach $\log_{m,a}(b)$ anstelle von $\log_{\mathbb{Z}_m^*,a}(b)$

Diskrete Logarithmen

- Die Funktion $\exp_{m,a} : x \mapsto a^x$ ist effizient berechenbar (siehe unten)
- Dagegen sind bis heute keine effizienten Verfahren zur Berechnung von $\log_{m,a}(b)$ bekannt (falls a und m geeignet gewählt werden)

Beispiel

- Das Element $a = 2$ hat in der Gruppe $G = \mathbb{Z}_{11}^*$ die maximal mögliche Ordnung $\mathrm{ord}_{11}(2) = \|G\| = 10$
- Die folgenden Tabellen zeigen den Werteverlauf der Funktionen $\exp_{11,2}$ und $\log_{11,2}$

x	0	1	2	3	4	5	6	7	8	9
2^x	1	2	4	8	5	10	9	7	3	6

b	1	2	3	4	5	6	7	8	9	10
$\log_{11,2}(b)$	0	1	8	2	4	9	7	3	6	5

◁

Zyklische Gruppen

Für manche Anwendungen sind Elemente $a \in G$ nützlich, mit denen sich die gesamte Gruppe erzeugen lässt

Definition

- Sei G eine endliche Gruppe der Ordnung $\|G\| = m$
- Ein Element $g \in G$ mit $\mathrm{ord}_G(g) = m$ heißt ***Erzeuger*** von G
- G heißt ***zyklisch***, falls G mindestens einen Erzeuger besitzt

Ein Element $a \in G$ ist also genau dann ein Erzeuger, wenn die von a erzeugte Untergruppe $\langle a \rangle$ die gesamte Gruppe G umfasst

Satz (Gauß)

Genau für $m \in \{1, 2, 4, p^k, 2p^k \mid 2 < p \text{ prim}\}$ ist die Gruppe $\mathbb{Z}_m^*$ zyklisch (ohne Beweis)

Das ElGamal-Signaturverfahren

- Das ***Signaturverfahren von ElGamal*** (1985) ist wie das gleichnamige asymmetrische Kryptosystem probabilistisch und beruht wie dieses auf dem diskreten Logarithmus
- Sei p eine große Primzahl und α ein Erzeuger von $\mathbb{Z}_p^*$
  (p und α sind öffentlich)
- Jeder Teilnehmer B wählt eine geheime Zahl $a \in \mathbb{Z}_{p-1} = \{0, \ldots, p-2\}$ und gibt $\beta = \alpha^a \bmod p$ als Teil seines öffentlichen Verifikationsschlüssels bekannt:

  Signierschlüssel: $\hat{k} = (p, \alpha, a)$
  Verifikationsschlüssel: $k = (p, \alpha, \beta)$
- Der Textraum ist $X = \mathbb{Z}_{p-1}$ und der Signaturenraum ist $Y = \mathbb{Z}_p^* \times \mathbb{Z}_{p-1} \setminus \{0\}$

Das ElGamal-Signaturverfahren

- Signaturerstellung: Um einen Text $x \in X$ zu signieren, wählt der Signierer zufällig eine Zahl $z \in \mathbb{Z}_{p-1}^*$ und berechnet die Signatur

  $$sig(\hat{k}, x, z) = (\gamma, \delta) \in Y$$

  mit $\gamma = \alpha^z \bmod p$ und $\delta = (x - a\gamma)z^{-1} \bmod p - 1$
- Falls $\delta = 0$ ist, muss eine neue Zufallszahl z gewählt und der Vorgang wiederholt werden
- Verifikation: Es gilt $ver(k, x, (\gamma, \delta)) = 1$, falls $\beta^\gamma \gamma^\delta \equiv_p \alpha^x$ ist

Das ElGamal-Signaturverfahren

Lemma

Eine Signatur (γ, δ) mit $\text{ord}(\gamma) = p - 1$ erfüllt genau dann die Verifikationsbedingung $\beta^\gamma \gamma^\delta \equiv_p \alpha^x$, wenn es ein $z \in \mathbb{Z}^*_{p-1}$ mit $sig(\hat{k}, x, z) = (\gamma, \delta)$ gibt

Beweis.

- Wegen $\gamma \equiv \alpha^z \bmod p$ ist z durch γ (und γ durch z) eindeutig bestimmt
- Weiter ist $\beta^\gamma \gamma^\delta \equiv_p \alpha^{a\gamma} \alpha^{z\delta} \equiv_p \alpha^{a\gamma + z\delta}$
- Da α ein Erzeuger von $\mathbb{Z}^*_p$ ist, gilt die Kongruenz $\alpha^{a\gamma+z\delta} \equiv_p \alpha^x$ genau dann, wenn $a\gamma + z\delta \equiv_{p-1} x$ ist, was wiederum mit $\delta \equiv_{p-1} (x - a\gamma)z^{-1}$ äquivalent ist

□

Bemerkung

Da der Signieralgorithmus für die Berechnung von $\gamma = \alpha^z \bmod p$ eine Zufallszahl $z \in \mathbb{Z}^*_{p-1}$ wählt, hat jedes von *sig* erzeugte γ die Ordnung $\text{ord}(\gamma) = \text{ord}(\alpha^z) = \text{ord}(\alpha)/\,\text{ggT}(\text{ord}(\alpha), z) = \text{ord}(\alpha) = p - 1$

Das ElGamal-Signaturverfahren

Beispiel

- Sei $p = 467$, $\alpha = 2$, $a = 127$ und $\beta = \alpha^a \bmod p = 2^{127} \bmod 467 = 132$
- Um den Text $x = 100 \in \mathbb{Z}_{p-1} = \mathbb{Z}_{466}$ mit dem Signierschlüssel $\hat{k} = (p, \alpha, a) = (467, 2, 127)$ zu signieren,
  - wählt Alice die geheime Zufallszahl $z = 213 \in \mathbb{Z}^*_{p-1}$ ($\rightsquigarrow z^{-1} \bmod 466 = 431$) und
  - erhält

    $$\gamma = 2^{213} \bmod 467 = 29 \text{ und } \delta = (100 - 127 \cdot 29)431 \bmod 466 = 51,$$

    d.h. $sig(\hat{k}, x, z) = (29, 51)$
- Um die Gültigkeit dieser Signatur für den Text $x = 100$ mit dem Verifikationsschlüssel $k = (p, \alpha, \beta) = (467, 2, 132)$ zu prüfen,
  - verifiziert Bob die Kongruenz

    $$\beta^\gamma \gamma^\delta \equiv_p 132^{29} 29^{51} \equiv_p 189 \equiv_p 2^{100} \equiv_p \alpha^x$$

Zur Sicherheit des ElGamal-Systems

- Falls der Angreifer in der Gruppe $\mathbb{Z}_p^*$ den diskreten Logarithmus von β zur Basis α bestimmen kann, so kann er den geheimen Schlüssel $a = \log_\alpha \beta$ berechnen
- Als nächstes betrachten wir verschiedene Szenarien für einen selektiven Angriff bei bekanntem Verifikationsschlüssel
- Der Angreifer wählt zu einem gegebenen Text x zuerst γ und versucht, ein passendes δ zu finden:
  - Mit $\alpha^x \equiv \beta^\gamma \gamma^\delta \bmod p$ folgt $\delta = \log_\gamma(\alpha^x \beta^{-\gamma})$
  - D.h. die Bestimmung von δ ist eine Instanz des diskreten Logarithmus Problems (kurz: DLP)
- Der Angreifer wählt zu einem gegebenen Text x zuerst δ und versucht dann ein γ mit $\alpha^x \equiv \beta^\gamma \gamma^\delta \bmod p$ zu finden
  - Hierfür ist kein effizientes Verfahren bekannt

Zur Sicherheit des ElGamal-Systems

- Der Angreifer versucht, zu einem gegebenen Text x gleichzeitig passende Zahlen γ und δ mit $\alpha^x \equiv \beta^\gamma \gamma^\delta \bmod p$ zu finden
  - Auch hierfür ist kein effizientes Verfahren bekannt
- Versucht der Angreifer bei einem nichtselektiven Angriff, zuerst γ und δ zu wählen und dazu einen passenden Text x zu finden, so muss er den diskreten Logarithmus $x = \log_\alpha \beta^\gamma \gamma^\delta$ bestimmen

Zur Sicherheit des ElGamal-Systems

- Eine existentielle Fälschung lässt sich jedoch wie folgt durchführen (falls keine Hashfunktion benutzt wird)
  - Der Angreifer wählt beliebige Zahlen $u \in \mathbb{Z}_{p-1}$, $v \in \mathbb{Z}^*_{p-1}$ und berechnet $\gamma = \alpha^u \beta^v \bmod p$
  - Dann ist (γ, δ) genau dann eine gültige Signatur für einen Text x, wenn $\alpha^x \equiv_p \beta^\gamma (\alpha^u \beta^v)^\delta$ ist
  - Dies ist wiederum äquivalent zur Kongruenz $\alpha^{x-u\delta} \equiv_p \beta^{\gamma+v\delta}$, die sich im Fall $\mathrm{ggT}(v, p-1) = 1$ für den Text $x = u\delta \bmod p-1$ mittels $\delta = -\gamma v^{-1} \bmod p-1$ erfüllen lässt
  - Bei Wahl von $v = 1$ erhalten wir z.B. die gültige Signatur $(\gamma, \delta) = (\alpha^u \beta \bmod p, -\alpha^u \beta \bmod p-1)$ für den Text $x = u\delta \bmod p-1$, wobei $u \in \mathbb{Z}_{p-1}$ beliebig gewählt werden kann

Zur Sicherheit des ElGamal-Systems

Bemerkung

Bei der Benutzung des ElGamal-Signaturverfahrens sind folgende Punkte zu beachten

- Die Zufallszahl z muss geheim gehalten werden
- Zufallszahlen dürfen nicht mehrfach verwendet werden

- Kennt nämlich der Angreifer zu einer Signatur $(x, (\gamma, \delta))$ die Zufallszahl z, so kann er wegen $\delta \equiv_{p-1} (x - a\gamma)z^{-1}$ im Fall $\mathrm{ggT}(\gamma, p-1) = 1$ die geheime Zahl

  $$a = (x - z\delta)\gamma^{-1} \bmod (p-1)$$

  als eindeutige Lösung der Kongruenz

  $$\gamma a \equiv_{p-1} x - z\delta \ (*)$$

  berechnen

Zur Sicherheit des ElGamal-Systems

- Kennt nämlich der Angreifer zu einer Signatur $(x, (\gamma, \delta))$ die Zufallszahl z, so kann er die geheime Zahl a als eindeutige Lösung der Kongruenz

  $$\gamma a \equiv_{p-1} x - z\delta \ (*)$$

  berechnen
- Ist allgemeiner $ggT(\gamma, p-1) = g \geq 1$, so ist g ein Teiler von γ und von $p-1$ sowie wegen $(*)$ auch von $x - z\delta$
- Setzen wir $\mu := \gamma/g$ und $\lambda := (x - z\delta)/g$, so führt $(*)$ auf die Kongruenz $\mu a \equiv_{(p-1)/g} \lambda \ (**)$, aus der sich wegen $ggT(\mu, (p-1)/g) = 1$ folgende g Kandidaten a_i für a gewinnen lassen:

  $a_0 := \mu^{-1}\lambda \bmod (p-1)/g$ und $a_i := a_0 + i(p-1)/g$ für $i = 1, \ldots, g-1$
- Unter $a_0, \ldots, a_{g-1}$ lässt sich a durch Prüfen der Bedingung $\alpha^{a_i} \equiv_p \beta$ eindeutig bestimmen

Zur Sicherheit des ElGamal-Systems

- Sind andererseits $(x_1, (\gamma, \delta_1))$ und $(x_2, (\gamma, \delta_2))$ mit demselben z generierte Signaturen, dann folgt wegen $\beta^\gamma \gamma^{\delta_i} \equiv_p \alpha^{x_i}$ für $i \in \{1, 2\}$,

$$\begin{aligned} \gamma^{\delta_1 - \delta_2} \equiv_p \alpha^{x_1 - x_2} \quad &\Rightarrow \quad \alpha^{z(\delta_1 - \delta_2)} \equiv_p \alpha^{x_1 - x_2} \\ &\Rightarrow \quad z(\delta_1 - \delta_2) \equiv_{p-1} x_1 - x_2 \end{aligned}$$

- Aus dieser Kongruenz lassen sich $d = ggT(\delta_1 - \delta_2, p - 1)$ Kandidaten für z gewinnen und daraus wie oben a berechnen, falls d nicht zu groß ist

Das Schnorr-Signaturverfahren

- Da die Primzahl p beim ElGamal-Signaturverfahren mindestens eine 512-Bit-Zahl (besser 1024-Bit-Zahl) sein sollte, beträgt die Signaturlänge 1024 bzw 2048 Bit
- Folgende Variante des ElGamal-Signaturverfahrens, die als eine Vorstufe zum DSA betrachtet werden kann, wurde von Schnorr vorgeschlagen
- Die zugrunde liegende Idee ist folgende:
  - Indem wir für α ein Element der Ordnung q mit $q \approx 2^{160}$ wählen, reduziert sich die Signaturlänge auf $2 \cdot 160 = 320$ Bit
  - Die Berechnungen werden aber nach wie vor modulo p mit $p \approx 2^{1024}$ ausgeführt, so dass das Problem des diskreten Logarithmus zur Basis α in $\mathbb{Z}_p^*$ hart bleibt

Das Schnorr-Signaturverfahren

- Sei g ein Erzeuger von $\mathbb{Z}_p^*$, wobei p die Bauart $p-1 = mq$ für eine Primzahl $q = \frac{p-1}{m} \approx 2^{160}$ hat
- Dann ist $\alpha = g^{(p-1)/q}$ ein Element in $\mathbb{Z}_p^*$ der Ordnung $ord_p(\alpha) = q$
  - da $ord(g^i) = \frac{ord(g)}{ggT(i,ord(g))} = \frac{p-1}{ggT((p-1)/q,p-1)} = q$ ist (s. Übungen)
- Weiter sei $h : \{0,1\}^* \to \mathbb{Z}_q$ eine Hashfunktion, die jedem Text $x \in X = \{0,1\}^*$ einen Hashwert in $\mathbb{Z}_q$ zuordnet
- Das Schnorr-Verfahren benutzt folgende Schlüssel:
  Signierschlüssel: $\hat{k} = (p, q, \alpha, a)$, $a \in \mathbb{Z}_q$
  Verifikationsschlüssel: $k = (p, \alpha, \beta)$, $\beta = \alpha^a \bmod p$

Das Schnorr-Signaturverfahren

- Das Schnorr-Verfahren benutzt folgende Schlüssel:
  Signierschlüssel: $\hat{k} = (p, q, \alpha, a)$, $a \in \mathbb{Z}_q$
  Verifikationsschlüssel: $k = (p, \alpha, \beta)$, $\beta = \alpha^a \bmod p$
- Signaturerstellung
  Um einen Text $x \in X$ zu signieren, wählt der Signierer zufällig eine geheime Zahl $z \in \mathbb{Z}_q^*$ (ElGamal: $z \in \mathbb{Z}_{p-1}^*$) und berechnet die Signatur

  $$sig(\hat{k}, x, z) = (\gamma, \delta),$$

  wobei $\gamma = h(xbin(\alpha^z \bmod p))$ und $\delta = (z + a\gamma) \bmod q$
  (ElGamal: $\gamma = \alpha^z \bmod p$ und $\delta = (x - a\gamma)z^{-1} \bmod p - 1$) ist
- Der Signaturraum ist also $Y := \mathbb{Z}_q \times \mathbb{Z}_q$
- Verifikation
  Es gilt $ver(k, x, \gamma, \delta) = 1$, falls $h(xbin(\alpha^\delta \beta^{-\gamma} \bmod p)) = \gamma$
  (ElGamal: $\beta^\gamma \gamma^\delta \equiv_p \alpha^x$) ist

Das Schnorr-Signaturverfahren

Beispiel

- Seien $q = 101$, $p = 78q + 1 = 7879$ und $g = 3$
- Dann ergibt sich α zu $\alpha = g^{(p-1)/q} = 3^{78} \bmod p = 170$
- Für $a = 75$ ergibt sich β zu $\beta = \alpha^a \bmod p = 170^{75} \bmod 7879 = 4567$
- Um einen Text $x \in \{0,1\}^*$ mit dem Signierschlüssel $\hat{k} = (p, q, \alpha, a) = (7879, 101, 170, 75)$ zu signieren,
  - wählt Alice die geheime Zufallszahl $z = 50 \in \mathbb{Z}_q^*$ und
  - berechnet den Wert $\alpha^z \bmod p = 170^{50} \bmod 7879 = 2518$
  - Dies führt auf den Hashwert $\gamma = h(x\mathit{bin}(2518)) \in \mathbb{Z}_q$
  - Unter der Annahme, dass $h(x\mathit{bin}(2518)) = 96$ ist, erhält Alice wegen

$$\delta = 50 + 75 \cdot 96 \bmod 101 = 79$$

die Signatur $\mathit{sig}(\hat{k}, x, z) = (96, 79)$

Das Schnorr-Signaturverfahren

Beispiel (Fortsetzung)

- Um die Gültigkeit der Signatur $sig(\hat{k}, x, z) = (96, 79)$ für den Text x mit dem Verifikationsschlüssel $k = (p, \alpha, \beta) = (7879, 170, 4567)$ zu prüfen,
  - berechnet Bob die Zahl

$$\alpha^{\delta}\beta^{-\gamma} \equiv_p 170^{79}4567^{-96} \equiv_p 2518$$

  - und verifiziert die Gleichheit $h(x\,bin(2518)) = 96$ ◁

Der Digital Signature Algorithm (DSA)

- Der DSA wurde im August 1991 vom National Institute of Standards and Technology (NIST) für die Verwendung im Digital Signature Standard (DSS) empfohlen
- Der DSS enthält neben dem DSA (ursprünglich der einzige im DSS definierte Algorithmus) als weitere Algorithmen die RSA-Signatur und ECDSA (siehe unten)
- Der DSA lässt sich durch eine Reihe von Modifikationen aus dem ElGamal-Verfahren erhalten, das wie folgt arbeitet

Der Digital Signature Algorithm (DSA)

- ElGamal-Verfahren:
  - Signaturerstellung: Um einen Text $x \in X$ zu signieren, wählt der Signierer zufällig eine Zahl $z \in \mathbb{Z}_{p-1}^*$ und berechnet die Signatur

    $$sig(\hat{k}, x, z) = (\gamma, \delta) \in Y$$

    mit $\gamma = \alpha^z \bmod p$ und $\delta = (x - a\gamma)z^{-1} \bmod p - 1$
  - Falls $\delta = 0$ ist, muss eine neue Zufallszahl z gewählt und der Vorgang wiederholt werden
  - Verifikation: Es gilt $ver(k, x, (\gamma, \delta)) = 1$, falls $\beta^\gamma \gamma^\delta \equiv_p \alpha^x$ ist
- Folge der Modifikationen für den Übergang zu DSA:
  - δ als Lösung von $z\delta - a\gamma \equiv_{p-1} x$ (d.h. $\delta = (x + a\gamma)z^{-1}$)
  - Dies führt auf die Verifikationsbedingung $\alpha^x \beta^\gamma \equiv_p \gamma^\delta$ $(\alpha^x \alpha^{a\gamma} \equiv_p \alpha^{z(x+a\gamma)z^{-1}})$
  - Ist $x + a\gamma \in \mathbb{Z}_{p-1}^*$, dann existiert $\delta^{-1} = (x + a\gamma)^{-1}z \bmod p - 1$
  - Dies führt auf die Verifikationsbedingung $\alpha^{x\delta^{-1}} \beta^{\gamma\delta^{-1}} \equiv_p \gamma$

Der Digital Signature Algorithm (DSA)

- Sei nun wie bei Schnorr $p = mq + 1$ mit $q \approx 2^{160}$ prim und sei $\alpha \in \mathbb{Z}_p^*$ mit $ord_p(\alpha) = q$
- Dann kann bei der Verifikation von $\alpha^{x\delta^{-1}} \beta^{\gamma\delta^{-1}} \equiv_p \gamma$ auf der Exponentenebene *modulo* q gerechnet werden
- Da γ jedoch rechts nicht als Exponent, sondern als Basiszahl, vorkommt, muss auch die linke Seite *modulo* q reduziert werden
- Beim DSA hat der Signierschlüssel also die Form $\hat{k} = (p, q, \alpha, a)$, wobei $a \in \mathbb{Z}_q^*$ ist
- Der zugehörige Verifikationsschlüssel ist $k = (p, q, \alpha, \beta)$ mit $\beta = \alpha^a \bmod p$
- Zudem gilt $X = \mathbb{Z}_q$ und $Y = \mathbb{Z}_q \times \mathbb{Z}_q^*$
- Zu gegebenem $x \in X$ wird zufällig eine geheime Zahl $z \in \mathbb{Z}_q^*$ gewählt

$$sig(\hat{k}, z, x) = (\gamma, \delta), \text{ wobei } \begin{cases} \gamma = (\alpha^z \bmod p) \bmod q \\ \delta = (x + a\gamma) z^{-1} \bmod q \in \mathbb{Z}_q^* \end{cases}$$

- Im Fall $\gamma = 0$ oder $\delta = 0$ muss ein neues z gewählt werden

Der Digital Signature Algorithm (DSA)

- Die Verifikationsbedingung ist

$$ver(k, x, \gamma, \delta) = \begin{cases} 1, & (\alpha^e \beta^d \bmod p) \bmod q = \gamma, \\ 0, & sonst, \end{cases}$$

  wobei $e = x\delta^{-1} \bmod q$ und $d = \gamma\delta^{-1} \bmod q$ ist
- Die Korrektheit ergibt sich wie folgt:
  - Im Fall $sig(\hat{k}, z, x) = (\gamma, \delta)$ ist

$$\alpha^e \beta^d \equiv_p \alpha^{x\delta^{-1}} \alpha^{a\gamma\delta^{-1}} \equiv_p \alpha^{\delta^{-1}(x+a\gamma)} \equiv_p \alpha^{(x+a\gamma)^{-1} z (x+a\gamma)} \equiv_p \alpha^z$$

    woraus sich

$$(\alpha^e \beta^d \bmod p) \bmod q = (\alpha^z \bmod p) \bmod q = \gamma$$

    ergibt

Der Digital Signature Algorithm (DSA)

Beispiel

- Seien $q = 101$, $p = 78q + 1 = 7879$, $g = 3 \quad (ord_p(3) = p - 1)$

  $$\rightsquigarrow \alpha = 3^{78} \bmod p = 170 \text{ hat Ordnung } q$$

- Wir wählen $a = 75 \in \mathbb{Z}_q^*$, d.h. $\beta = \alpha^a \bmod p = 170^{75} \bmod p = 4567$
- Um den Text $x = 22 \in \mathbb{Z}_q$ zu signieren, wählen wir die geheime Zufallszahl $z = 50 \in \mathbb{Z}_q^*$ ($\rightsquigarrow z^{-1} = 99$) und erhalten dann

$$\begin{aligned} \gamma &= (170^{50} \bmod 7879) \bmod 101 \\ &= 2518 \bmod 101 \\ &= 94 \\ \delta &= (22 + 75 \cdot 94) \cdot 99 \bmod 101 \\ &= 97 \ (\rightsquigarrow \delta^{-1} = 25) \end{aligned}$$

d.h. $sig(p, q, \alpha, z, x) = (94, 97)$, wobei $\hat{k} = (p, q, \alpha, a)$

Der Digital Signature Algorithm (DSA)

Beispiel (Fortsetzung)

- Um diese Signatur zu prüfen berechnen wir:

$$\begin{aligned} e &= x\delta^{-1} \bmod q \\ &= 22 \cdot 25 \bmod 101 \\ &= 45 \\ d &= \gamma\delta^{-1} \bmod q \\ &= 94 \cdot 25 \bmod 101 \\ &= 27 \end{aligned}$$

$\rightsquigarrow (\alpha^e \beta^d \bmod p) \bmod q = (170^{45} 4547^{27} \bmod 7879) \bmod 101 = 94$ ◁

Der ECDSA (Elliptic Curve DSA)

- Der ECDSA wurde im Jahr 2000 als FIPS (Federal Information Processing Standard) 186-2 Standard deklariert
- Sei E eine elliptische Kurve über einem endlichen Körper $\mathbb{F}_{p^n}$
- Sei $A \in E$ ein Punkt der Ordnung q (q prim), so dass das Diskrete-Logarithmus-Problem zur Basis A in E schwierig ist
- Zudem sei $h\colon \{0,1\}^* \to \mathbb{Z}_q$ eine kryptografische Hashfunktion
- Der ECDSA besteht aus folgenden Komponenten:

  Textraum: $X = \{0,1\}^*$
  Signaturraum: $Y = \mathbb{Z}_q^* \times \mathbb{Z}_q^*$
  Signierschlüssel: $\hat{k} = (E, q, A, m)$, $m \in \mathbb{Z}_q^*$
  Verifikationsschlüssel: $k = (E, q, A, B)$, wobei $B = m \cdot A$ ist

Der ECDSA (Elliptic Curve DSA)

- Signaturerstellung: Um einen Text $x \in X$ zu signieren,
  - wählt der Signierer zufällig eine geheime Zahl $z \in \mathbb{Z}_q^*$ und
  - berechnet $sig(\hat{k}, x, z) = (\gamma, \delta)$ mit

$$\begin{aligned} (u, v) &:= zA \\ \gamma &:= u \bmod q \\ \delta &:= (h(x) + m\gamma)z^{-1} \bmod q \end{aligned}$$

- Hierbei wird u als eine Zahl in $\{0, \ldots, p^n - 1\}$ interpretiert
- Falls $\gamma = 0$ oder $\delta = 0$ ist, muss eine neue Zufallszahl z gewählt und der Vorgang wiederholt werden

Der ECDSA (Elliptic Curve DSA)

- Verifikation: $ver(k, x, \gamma, \delta) = 1$, falls $u \bmod q = \gamma$ ist, wobei

$$\begin{aligned} e &:= h(x)\delta^{-1} \bmod q \\ d &:= \gamma\delta^{-1} \bmod q \\ (u, v) &:= eA + dB \end{aligned}$$

- Korrektheit der Verifikation beim ECDSA:

$$\begin{aligned} (u, v) &= eA + dB \\ &= (h(x)\delta^{-1})A + (\gamma\delta^{-1})mA \\ &= (h(x) + m\gamma)\delta^{-1}A \\ &= zA \text{ (da } (h(x) + m\gamma)\delta^{-1} \equiv_q z) \end{aligned}$$

Der ECDSA (Elliptic Curve DSA)

Beispiel

- Sei E über $\mathbb{Z}_{11}$ definiert durch $y^2 = x^3 + x + 6$
- Wir wählen $A = (2, 7)$, $m = 7 \rightarrow p = 11, q = 13, B = 7A = (7, 2)$
- Um einen Text x mit dem Hashwert $h(x) = 4$ unter Verwendung des Signierschlüssels $\hat{k} = (E, q, A, m)$ und der Zufallszahl $z = 3$ signieren,
  - berechnet Alice

$$\begin{aligned} (u, v) &:= zA = 3 \cdot (2, 7) = (8, 3) \\ \gamma &:= u \bmod q = 8 \\ \delta &:= (4 + 7 \cdot 8)3^{-1} \bmod 13 = 7 \end{aligned}$$

  - und erhält die Signatur $sig(\hat{k}, z, x) = (8, 7)$

Der ECDSA (Elliptic Curve DSA)

Beispiel (Fortsetzung)

- Um diese Signatur mit dem Verifikationsschlüssel $k = (E, q, A, B)$ zu überprüfen,
  - berechnet Bob

$$\begin{aligned} e &:= h(x)\delta^{-1} \bmod q = 4 \cdot 7^{-1} \bmod 13 = 4 \cdot 2 \bmod 13 = 8 \\ d &:= \gamma\delta^{-1} \bmod q = 8 \cdot 2 \bmod 13 = 3 \\ (u, v) &:= eA + dB = 8 \cdot (2, 7) + 3 \cdot (7, 2) = (8, 3) \end{aligned}$$

  - und testet die Kongruenz $u \equiv_q \gamma$ ◁

Die One-time-Signatur von Lamport

- Leslie Lamport konnte 1979 zeigen, dass sich digitale Signaturen auf der Basis einer Einwegfunktion f konstruieren lassen
- Damit die Signatur allerdings sicher ist, muss für jeden Text ein neues Schlüsselpaar $(\hat{k}, k)$ generiert werden
- Ein Signierschlüssel $\hat{k}$ darf also nur zum Signieren eines einzelnen Textes verwendet werden
- Seien U und V endliche Mengen und sei $f : U \to V$ eine Funktion
- Zudem sei $\ell \geq 1$ die vorgegebene Textlänge, d.h. der Textraum ist $X = \{0,1\}^\ell$
- Der Signaturraum ist dann $Y = U^\ell$
- Um ein Schlüsselpaar $(\hat{k}, k)$ zu generieren, wird zufällig eine Folge von 2ℓ Elementen $u_{i,b}$ für $i = 1, \ldots, \ell$ und $b = 0, 1$ aus U gewählt und der Signierschlüssel $\hat{k} = \binom{u_{1,0} \ldots u_{\ell,0}}{u_{1,1} \ldots u_{\ell,1}}$ gebildet
- Der zugehörige Verifikationsschlüssel ist dann $k = \binom{v_{1,0} \ldots v_{\ell,0}}{v_{1,1} \ldots v_{\ell,1}}$ mit $v_{i,b} = f(u_{i,b})$ für alle $i = 1, \ldots, \ell$ und $b = 0, 1$

Die One-time-Signatur von Lamport

- Signaturerstellung: Die Signatur für einen Text $x = x_1 \dots x_\ell \in X$ ist

$$sig(\hat{k}, x) = (u_{1,x_1}, \dots, u_{\ell,x_\ell})$$

- Verifikation: Für eine Signatur $y = (u_1, \dots, u_\ell)$ und einen Text $x = x_1 \dots x_\ell$ gilt

$$ver(k, x, y) = \begin{cases} 1, & f(u_i) = v_{i,x_i} \text{ für } i = 1, \dots, \ell, \\ 0, & sonst \end{cases}$$

Die One-time-Signatur von Lamport

Beispiel

- Wir wählen als Einwegfunktion eine Funktion der Form $f : \mathbb{Z}_p^* \to \mathbb{Z}_p^*$ mit $f(u) = g^u \bmod p$, wobei g ein Erzeuger von $\mathbb{Z}_p^*$ ist
- Z.B. sei $p = 7879$ und $g = 3$, also $f(u) = 3^u \bmod 7879$
- Weiter sei $\ell = 3$
- Dann erhalten wir für den zufällig gewählten Signierschlüssel $\hat{k} = \begin{pmatrix} 5831 & 4285 & 2467 \\ 803 & 735 & 6449 \end{pmatrix}$ den Verifikationsschlüssel $k = \begin{pmatrix} 2009 & 268 & 4721 \\ 4672 & 3810 & 5731 \end{pmatrix}$
- Die Signatur y für den Text $x = 110$ ist dann

$$y = sig(\hat{k}, x) = (u_{1,x_1}, u_{2,x_2}, u_{3,x_3}) = (u_{1,1}, u_{2,1}, u_{3,0}) = (803, 735, 2467)$$

- Für diese Signatur $y = (u_1, u_2, u_3)$ ist $ver(k, x, y) = 1$, da $f(u_i) = v_{i,x_i}$ für $i = 1, 2, 3$ gilt:

$$i = 1 : f(u_1) = f(803) = 3^{803} \bmod 7879 = 4672 = v_{1,x_1}$$
$$i = 2 : f(u_2) = f(735) = 3^{735} \bmod 7879 = 3810 = v_{2,x_2}$$
$$i = 3 : f(u_3) = f(2467) = 3^{2467} \bmod 7879 = 4721 = v_{3,x_3}$$

◁

Die One-time-Signatur von Lamport

- Ähnlich wie bei MACs können wir einen Angriff gegen ein digitales Signaturverfahren wie folgt modellieren
- Hierbei nehmen wir an, dass der Angreifer die Texte, deren Signaturen er kennt, adaptiv wählen kann
- Es handelt sich also um eine existentielle Fälschung bei adaptiv wählbaren Texten

Definition. Sei $0 \leq \varepsilon \leq 1$ und sei $q \in \mathbb{N}$

- Ein (ε, q)-***Fälscher*** für ein digitales Signaturverfahren ist ein probabilistischer Algorithmus $\mathcal{A}$, der
  - bei Eingabe eines Verifikationsschlüssels k, wobei das Schlüsselpaar $(\hat{k}, k)$ zufällig gewählt wird
  - nach den Signaturen $y_i = sig(\hat{k}, x_i)$ von q Texten $x_1, \ldots, x_q$ adaptiv fragt und
  - mit Wahrscheinlichkeit mindestens ε eine Fälschung (x, y) mit $x \notin \{x_1, \ldots, x_q\}$ und $ver(k, x, y) = 1$ ausgibt

Die One-time-Signatur von Lamport

Satz. Sei $f : U \to V$ eine Funktion

Falls für die zugehörige one-time Signatur ein $(\varepsilon, 0)$-Fälscher $\textsc{Lamport-Fälschung}(k)$ existiert, dann lässt sich für ein zufällig gewähltes $u \in_R U$ mit Wahrscheinlichkeit mindestens $\varepsilon/2$ ein Urbild von $v = f(u)$ bestimmen

Beweis.

Betrachte folgenden probabilistischen Algorithmus $\textsc{Lamport-Urbild}(v)$:

Prozedur `Lamport-Urbild`(v)

1. wähle zufällig ein Indexpaar (j, a) und setze $v_{j,a} := v$
2. **for all** $(i, b) \in [\ell] \times \{0, 1\} \setminus \{(j, a)\}$ **do**
3. wähle zufällig $u_{i,b} \in_R U$ und setze $v_{i,b} := f(u_{i,b})$
4. $k := \binom{v_{1,0} \ldots v_{\ell,0}}{v_{1,1} \ldots v_{\ell,1}}$
5. $(x_1 \ldots x_\ell, (u_1, \ldots, u_\ell)) =: \textsc{Lamport-Fälschung}(k)$
6. **if** $f(u_j) = v$ **then output**(u_j) **else output**$(?)$

Die One-time-Signatur von Lamport

Beweis (Fortsetzung)

- Wie üblich bezeichnen wir die Zufallsvariablen, die die Wahl von v, j, a, k und $(x, y) = (x_1 \dots x_\ell, (u_1, \dots, u_\ell))$ beschreiben, mit entsprechenden Großbuchstaben
- Dann müssen wir zeigen, dass U_J mit Wahrscheinlichkeit mindestens $\varepsilon/2$ ein f-Urbild von V ist, wobei V die Wahl von $v = f(u)$ für ein zufällig gewähltes $u \in_R U$ beschreibt
- Da die Verteilung von K identisch zur Schlüsselgenerierung der Lamport-Signatur und $\textsc{Lamport-Fälschung}$ ein $(\varepsilon, 0)$-Fälscher ist, folgt

$$\Pr[ver(K, X, Y) = 1] \geq \varepsilon$$

- Da zudem K (und damit auch (X, Y)) unabhängig von (J, A) und auch J und A unabhängig voneinander sind, ist A von (J, K, X, Y) und damit auch von X_J unabhängig

Die One-time-Signatur von Lamport

Beweis (Schluss)

- Sei p die Erfolgswk von LAMPORT-URBILD bei Eingabe V
- Wegen

$$ver(k, x_1 \dots x_\ell, (u_1, \dots, u_\ell)) = 1 \wedge x_j = a \quad \Rightarrow \quad f(u_j) = v_{j,x_j} = v_{j,a} = v$$

folgt nun

$$\begin{aligned} p &\geq \Pr[ver(K, X, Y) = 1 \wedge X_J = A] \\ &= \underbrace{\Pr[ver(K, X, Y) = 1]}_{\geq \varepsilon} \underbrace{\Pr[X_J = A \mid ver(K, X, Y) = 1]}_{=1/2} \\ &\geq \varepsilon/2 \end{aligned}$$

□

Die One-time-Signatur von Lamport

Als nächstes untersuchen wir die Sicherheit der Lamport-Signatur, falls der Angreifer in der Lage ist, sich für einen beliebigen Text x' seiner Wahl eine gültige Signatur y' zu beschaffen

Satz. Sei $f : U \to V$ eine Funktion.

Falls für die zugehörige one-time Signatur ein $(\varepsilon, 1)$-Fälscher $\textsc{Lamport-Fälschung'}(k)$ existiert, so lässt sich für ein zufällig gewähltes $u \in_R U$ mit Wahrscheinlichkeit $\geq \varepsilon/2\ell$ ein f-Urbild von $v = f(u)$ bestimmen

Für den Beweis betrachten wir folgenden probabilistischen Algorithmus `Lamport-Urbild'` und zeigen, dass er für ein zufällig gewähltes $u \in_R U$ bei Eingabe $v = f(u)$ mit Wahrscheinlichkeit $\geq \varepsilon/2\ell$ ein f-Urbild von v ausgibt

Die One-time-Signatur von Lamport

Für den Beweis betrachten wir folgenden probabilistischen Algorithmus `Lamport-Urbild'` und zeigen, dass er für ein zufällig gewähltes $u \in_R U$ bei Eingabe $v = f(u)$ mit Wahrscheinlichkeit $\geq \varepsilon/2\ell$ ein f-Urbild von v ausgibt:

Prozedur `Lamport-Urbild'`(v)

1 wähle zufällig ein Indexpaar (j, a) und setze $v_{j,a} := v$
2 **for all** $(i, b) \neq (j, a)$ **do**
3 wähle zufällig $u_{i,b} \in_R U$ und setze $v_{i,b} := f(u_{i,b})$
4 $k := \binom{v_{1,0} \ldots v_{\ell,0}}{v_{1,1} \ldots v_{\ell,1}}$
5 simuliere LAMPORT-FÄLSCHUNG'(k) und beantworte die Frage x' mit $u_{1,x'_1}, \ldots, u_{\ell,x'_\ell}$ (falls $x'_j = a$ ist, brich ab und gib ? aus);
6 sei $(x, y) = (x_1 \ldots x_\ell, (u_1, \ldots, u_\ell))$ die erzeugte Ausgabe
7 **if** $f(u_j) = v$ **then output**(u_j) **else output**$(?)$

Die One-time-Signatur von Lamport

Beweis.

- Sei p' die Erfolgswk von LAMPORT-URBILD' bei Eingabe V
- LAMPORT-URBILD' kann die Frage von LAMPORT-FÄLSCHUNG'(k) nach der Signatur von x' nur dann beantworten, wenn $x'_j \neq a$ ist
- Es ist klar, dass in diesem Fall u_j ein Urbild von v ist, wenn zudem $ver(k, x_1 \dots x_\ell, (u_1, \dots, u_\ell)) = 1 \wedge x_j = a$ gilt
- Da jedoch die Simulation von LAMPORT-FÄLSCHUNG'(k) eventuell abgebrochen wird (und die Abbruchbedingung von (j, a) abhängt), können wir nicht mehr davon ausgehen, dass diese Simulation mit Wahrscheinlichkeit ε eine Fälschung (x, y) liefert und (X, Y) unabhängig von (J, A) ist
- Durch eine einfache Modifikation von LAMPORT-URBILD'(v) erhalten wir jedoch eine Prozedur LAMPORT-URBILD* (ohne Eingabe), deren Ausgabeverhalten mit der von LAMPORT-URBILD'(V) identisch ist, und von der wir zeigen können, dass sie mit Wahrscheinlichkeit $p^* \geq \varepsilon/2\ell$ Erfolg hat (also nicht Fragezeichen ausgibt):

Die One-time-Signatur von Lamport

Beweis (Fortsetzung)

- Durch eine einfache Modifikation von LAMPORT-URBILD'(v) erhalten wir jedoch eine Prozedur LAMPORT-URBILD* (ohne Eingabe), deren Ausgabeverhalten mit der von LAMPORT-URBILD'(V) identisch ist, und von der wir zeigen können, dass sie mit Wahrscheinlichkeit $p^* \geq \varepsilon/2\ell$ Erfolg hat (also nicht Fragezeichen ausgibt):

Prozedur `Lamport-Urbild*`

1 wähle zufällig ein Indexpaar (j, a)
2 **for all** (i, b) **do** wähle zufällig $u_{i,b} \in_R U$ und setze $v_{i,b} := f(u_{i,b})$
3 $k := \binom{v_{1,0} \ldots v_{\ell,0}}{v_{1,1} \ldots v_{\ell,1}}$
4 simuliere LAMPORT-FÄLSCHUNG'(k) und beantworte die Frage x' mit $u_{1,x'_1}, \ldots, u_{\ell,x'_\ell}$;
5 sei $(x, y) = (x_1 \ldots x_\ell, (u_1, \ldots, u_\ell))$ die erzeugte Ausgabe
6 **if** $f(u_j) = v_{j,a} \wedge x'_j \neq a$ **then output**(u_j) **else output**$(?)$

Die One-time-Signatur von Lamport

Beweis (Fortsetzung)

- Im Unterschied zu LAMPORT-URBILD'(v) wählt sich LAMPORT-URBILD* also die Eingabe $v = v_{j,a}$ gemäß der Verteilung von V selbst und kennt daher auch ein Urbild $u_{j,a}$ von $v_{j,a}$
- Somit kann LAMPORT-URBILD* bei der Simulation von LAMPORT-FÄLSCHUNG'(k) die Frage nach der Signatur von x' auch im Fall $x'_j = a$ beantworten
- Die Bedingung für die Ausgabe von u_j ist jedoch bei beiden Prozeduren dieselbe, d.h. die Ausgabe von LAMPORT-URBILD* hat dieselbe Verteilung wie die von LAMPORT-URBILD'(V) und somit gilt $p' = p^*$
- Der einzige Unterschied ist, dass immer wenn LAMPORT-URBILD'(V) in Zeile 5 ein Fragezeichen ausgibt, LAMPORT-URBILD* dies erst in Zeile 6 tut

Die One-time-Signatur von Lamport

Beweis (Schluss)

- Da in der Prozedur LAMPORT-URBILD* die ZV (J, A) unabhängig von (K, X', X, Y) ist, folgt nun

$$\begin{aligned} p* &= \Pr[f(U_J) = V_{J,A} \wedge X'_J \neq A] \\ &\geq \Pr[ver(K, X, Y) = 1 \wedge X_J = A \wedge X'_J \neq A] \\ &= \Pr[ver(K, X, Y) = 1] \underbrace{\Pr[X'_J \neq X_J = A \mid ver(K, X, Y) = 1]}_{\geq 1/2\ell} \\ &\geq \varepsilon/2\ell \end{aligned}$$

□

- Die Lamport-Signatur hat aus praktischer Sicht einige Nachteile, die sich jedoch teilweise beheben lassen (siehe Übungen)
- So lässt sich sowohl die Länge des privaten Signierschlüssels (mittels Pseudozufallsgeneratoren) als auch des öffentlichen Verifikationsschlüssels (mittels Hash-Listen) verringern
- Zudem können bei Verwendung von Hash-Bäumen mit demselben Schlüsselpaar auch mehrere Nachrichten signiert und verifiziert werden